Lecture Notes
in Economics and
Mathematical Systems

Managing Editors: M. Beckmann and W. Krelle

304

A. Kurzhanski K. Neumann
D. Pallaschke (Eds.)

Optimization, Parallel Processing and Applications

Proceedings, Oberwolfach and Karlsruhe
West Germany, 1987

Springer-Verlag

Lecture Notes in Economics and Mathematical Systems

For information about Vols. 1–156, please contact your bookseller or Springer-Verlag

Vol. 157: Optimization and Operations Research. Proceedings 1977. Edited by R. Henn, B. Korte, and W. Oettli. VI, 270 pages. 1978.

Vol. 158: L. J. Cherene, Set Valued Dynamical Systems and Economic Flow. VIII, 83 pages. 1978.

Vol. 159: Some Aspects of the Foundations of General Equilibrium Theory: The Posthumous Papers of Peter J. Kalman. Edited by J. Green. VI, 167 pages. 1978.

Vol. 160: Integer Programming and Related Areas. A Classified Bibliography. Edited by D. Hausmann. XIV, 314 pages. 1978.

Vol. 161: M. J. Beckmann, Rank in Organizations. VIII, 164 pages. 1978.

Vol. 162: Recent Developments in Variable Structure Systems, Economics and Biology. Proceedings 1977. Edited by R. R. Mohler and A. Ruberti. VI, 326 pages. 1978.

Vol. 163: G. Fandel, Optimale Entscheidungen in Organisationen. VI, 143 Seiten. 1979.

Vol. 164: C. L. Hwang and A. S. M. Masud, Multiple Objective Decision Making – Methods and Applications. A State-of-the-Art Survey. XII, 351 pages. 1979.

Vol. 165: A. Maravall, Identification in Dynamic Shock-Error Models. VIII, 158 pages. 1979.

Vol. 166: R. Cuninghame-Green, Minimax Algebra. XI, 258 pages. 1979.

Vol. 167: M. Faber, Introduction to Modern Austrian Capital Theory. X, 196 pages. 1979.

Vol. 168: Convex Analysis and Mathematical Economics. Proceedings 1978. Edited by J. Kriens. V, 136 pages. 1979.

Vol. 169: A. Rapoport et al., Coalition Formation by Sophisticated Players. VII, 170 pages. 1979.

Vol. 170: A. E. Roth, Axiomatic Models of Bargaining. V, 121 pages. 1979.

Vol. 171: G. F. Newell, Approximate Behavior of Tandem Queues. XI, 410 pages. 1979.

Vol. 172: K. Neumann and U. Steinhardt, GERT Networks and the Time-Oriented Evaluation of Projects. 268 pages. 1979.

Vol. 173: S. Erlander, Optimal Spatial Interaction and the Gravity Model. VII, 107 pages. 1980.

Vol. 174: Extremal Methods and Systems Analysis. Edited by A. V. Fiacco and K. O. Kortanek. XI, 545 pages. 1980.

Vol. 175: S. K. Srinivasan and R. Subramanian, Probabilistic Analysis of Redundant Systems. VII, 356 pages. 1980.

Vol. 176: R. Färe, Laws of Diminishing Returns. VIII, 97 pages. 1980.

Vol. 177: Multiple Criteria Decision Making-Theory and Application. Proceedings, 1979. Edited by G. Fandel and T. Gal. XVI, 570 pages. 1980.

Vol. 178: M. N. Bhattacharyya, Comparison of Box-Jenkins and Bonn Monetary Model Prediction Performance. VII, 146 pages. 1980.

Vol. 179: Recent Results in Stochastic Programming. Proceedings, 1979. Edited by P. Kall and A. Prékopa. IX, 237 pages. 1980.

Vol. 180: J. F. Brotchie, J. W. Dickey and R. Sharpe, TOPAZ – General Planning Technique and its Applications at the Regional, Urban, and Facility Planning Levels. VII, 356 pages. 1980.

Vol. 181: H. D. Sherali and C. M. Shetty, Optimization with Disjunctive Constraints. VIII, 156 pages. 1980.

Vol. 182: J. Wolters, Stochastic Dynamic Properties of Linear Econometric Models. VIII, 154 pages. 1980.

Vol. 183: K. Schittkowski, Nonlinear Programming Codes. VIII, 242 pages. 1980.

Vol. 184: R. E. Burkard and U. Derigs, Assignment and Matching Problems: Solution Methods with FORTRAN-Programs. VIII, 148 pages. 1980.

Vol. 185: C. C. von Weizsäcker, Barriers to Entry. VI, 220 pages. 1980.

Vol. 186: Ch.-L. Hwang and K. Yoon, Multiple Attribute Decision Making – Methods and Applications. A State-of-the-Art-Survey. XI, 259 pages. 1981.

Vol. 187: W. Hock, K. Schittkowski, Test Examples for Nonlinear Programming Codes. V. 178 pages. 1981.

Vol. 188: D. Bös, Economic Theory of Public Enterprise. VII, 142 pages. 1981.

Vol. 189: A. P. Lüthi, Messung wirtschaftlicher Ungleichheit. IX, 287 pages. 1981.

Vol. 190: J. N. Morse, Organizations: Multiple Agents with Multiple Criteria. Proceedings, 1980. VI, 509 pages. 1981.

Vol. 191: H. R. Sneessens, Theory and Estimation of Macroeconomic Rationing Models. VII, 138 pages. 1981.

Vol. 192: H. J. Bierens: Robust Methods and Asymptotic Theory in Nonlinear Econometrics. IX, 198 pages. 1981.

Vol. 193: J. K. Sengupta, Optimal Decisions under Uncertainty. VII, 156 pages. 1981.

Vol. 194: R. W. Shephard, Cost and Production Functions. XI, 104 pages. 1981.

Vol. 195: H. W. Ursprung, Die elementare Katastrophentheorie. Eine Darstellung aus der Sicht der Ökonomie. VII, 332 pages. 1982.

Vol. 196: M. Nermuth, Information Structures in Economics. VIII, 236 pages. 1982.

Vol. 197: Integer Programming and Related Areas. A Classified Bibliography. 1978 – 1981. Edited by R. von Randow. XIV, 338 pages. 1982.

Vol. 198: P. Zweifel, Ein ökonomisches Modell des Arztverhaltens. XIX, 392 Seiten. 1982.

Vol. 199: Evaluating Mathematical Programming Techniques. Proceedings, 1981. Edited by J.M. Mulvey. XI, 379 pages. 1982.

Vol. 200: The Resource Sector in an Open Economy. Edited by H. Siebert. IX, 161 pages. 1984.

Vol. 201: P. M. C. de Boer, Price Effects in Input-Output-Relations: A Theoretical and Empirical Study for the Netherlands 1949–1967. X, 140 pages. 1982.

Vol. 202: U. Witt, J. Perske, SMS – A Program Package for Simulation and Gaming of Stochastic Market Processes and Learning Behavior. VII, 266 pages. 1982.

Vol. 203: Compilation of Input-Output Tables. Proceedings, 1981. Edited by J. V. Skolka. VII, 307 pages. 1982.

Vol. 204: K. C. Mosler, Entscheidungsregeln bei Risiko: Multivariate stochastische Dominanz. VII, 172 Seiten. 1982.

Vol. 205: R. Ramanathan, Introduction to the Theory of Economic Growth. IX, 347 pages. 1982.

Vol. 206: M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Redundancy in Mathematical Programming. VII, 286 pages. 1983.

Vol. 207: Y. Fujimori, Modern Analysis of Value Theory. X, 165 pages. 1982.

Vol. 208: Econometric Decision Models. Proceedings, 1981. Edited by J. Gruber. VI, 364 pages. 1983.

Vol. 209: Essays and Surveys on Multiple Criteria Decision Making. Proceedings, 1982. Edited by P. Hansen. VII, 441 pages. 1983.

Vol. 210: Technology, Organization and Economic Structure. Edited by R. Sato and M. J. Beckmann. VIII, 195 pages. 1983.

continuation on page 297

Lecture Notes in Economics and Mathematical Systems

Managing Editors: M. Beckmann and W. Krelle

304

A. Kurzhanski K. Neumann
D. Pallaschke (Eds.)

Optimization, Parallel Processing and Applications

Proceedings of the Oberwolfach Conference on Operations Research, February 16–21, 1987 and the Workshop on Advanced Computation Techniques, Parallel Processing and Optimization
Held at Karlsruhe, West Germany, February 22–25, 1987

Spri

Berlin

Editorial Board

H. Albach M. Beckmann (Managing Editor) P. Dhrymes
G. Fandel G. Feichtinger J. Green W. Hildenbrand W. Krelle (Managing Editor)
H.P. Künzi K. Ritter R. Sato U. Schittko P. Schönfeld R. Selten

Managing Editors

Prof. Dr. M. Beckmann
Brown University
Providence, RI 02912, USA

Prof. Dr. W. Krelle
Institut für Gesellschafts- und Wirtschaftswissenschaften
der Universität Bonn
Adenauerallee 24–42, D-5300 Bonn, FRG

Editors

Prof. Dr. Alexander Kurzhanski
International Institute for Applied Systems Analysis (IIASA)
Schloßplatz 1, A-2361 Laxenburg, Austria
and
Institute of Mathematics and Mechanics of the Ural Scientific Center
of the Academy of Sciences of the USSR
Sverdlovsk, USSR

Prof. Dr. Klaus Neumann
Institut für Wirtschaftstheorie und Operations Research
Universität Karlsruhe
Kaiserstr. 12, D-7500 Karlsruhe, FRG

Prof. Dr. Diethard Pallaschke
Institut für Statistik und Mathematische Wirtschaftstheorie
Universität Karlsruhe
Postfach 6980, D-7500 Karlsruhe 1, FRG

ISBN-13: 978-3-540-19053-0 e-ISBN-13: 978-3-642-46631-1
DOI: 10.1007/978-3-642-46631-1

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this publication or parts thereof is only permitted under the provisions of the German Copyright Law of September 9, 1965, in its version of June 24, 1985, and a copyright fee must always be paid. Violations fall under the prosecution act of the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1988

2145/3140-543210

PREFACE

It is already a tradition that conferences on operations research are organized by the Mathematisches Forschungsinstitut in Oberwolfach/Germany.
The mean point of the 1987 conference was to discuss recently developed methods in optimization theory derived from various fields of mathematics. On the other hand, the practical use of results in operations research is very important. In the last few years essential progress in this direction was made at the International Institute for Applied Systems Analysis (IIASA) at Laxenburg/Austria. Therefore a three days workshop on Advanced Computation Techniques, Parallel Processing and Optimization organized by IIASA and the University of Karlsruhe immediately followed the Oberwolfach Conference.

This volume contains selected papers which have been presented at one of these conferences. It is divided into five sections based on the above topics:

I. Algorithms and Optimization Methods

II. Optimization and Parallel Processing

III. Graph Theory and Scheduling

IV. Differential Equations and Operator Theory

V. Applications

We would like to thank the director of the Mathematisches Forschungsinstitut Oberwolfach Prof. Dr. M. Barner and the International Institute for Applied Systems Analysis, particularly Prof. Dr. V. Kaftanov, and also to the director of the Computer Center of the University of Karlsruhe Prof. Dr. A. Schreiner for their support in organizing these conferences. We also appreciate the excellent cooperation of Springer Verlag.

We also thank Dr. P. Recht, Dr. D. Solte and Dr. K. Wieder as well as Mrs. Dagmar Hoffmann, Korina Recht and Monika Schoppa for their help during these conferences.

A. Kurzhanski
K. Neumann
D. Pallaschke

CONTENTS

I. ALGORITHMS AND OPTIMIZATION METHODS

Quantitative Stability of (Mixed-)Integer Linear Optimization Problems 3
B. Bank and R. Mandel (GDR)

Parallel Hybrid Optimization Methods 16
O. Burdakov (USSR) and C. Richter (GDR)

Continuous Generalized Gradients for Nonsmooth Functions 24
V. F. Demyanov (USSR)

Outer Cut Methods in Global Optimization 28
R. Horst (FRG)

Generic Uniqueness of the Solution of "Max Min" Problems 41
P. S. Kenderov and N. K. Ribarska (Bulgaria)

Optimal Stopping and Leavable Gambling Models with Observation Costs 49
M. Schäl (FRG)

Multi-Ratio Fractional Programming - A Survey 57
S. Schaible (Canada)

II. OPTIMIZATION AND PARALLEL PROCESSING

An Experience of Advanced Computation Techniques in the Solution of Nonlinearly Constrained Optimization Problems 69
D. Conforti and L. Grandinetti (Italy)

Automatic Differentiation and Parallel Processing in Optimisation 86
L. C. W. Dixon (Great Britain)

Multicriteria Optimization in the DISO System 94
Y. Evtushenko, V. Mazourik and V. Ratkin (USSR)

Serial and Parallel Solution of Large Scale Linear Programs by Augmented Lagrangian Successive Overrelaxation 103
R. De Leone and O. L. Mangasarian (USA)

EMP: An Expert System for Mathematical Programming 125
K. Schittkowski (FRG)

Effective Algorithm for Global Optimization with Parallel Computations 158
R. G. Strongin and Y. D. Sergeev (USSR)

III. GRAPH THEORY AND SCHEDULING

M-Machine Unit Time Scheduling: A Report on Ongoing Research 165
M. Bartusch, R. H. Möhring and F. J. Radermacher (FRG)

On Inverse-Image of Non-Oriented Graphs 213
S. Perz and S. Rolewicz (Poland)

IV. DIFFERENTIAL EQUATIONS AND OPERATOR THEORY

Existence of Optimal Trajectory of Mayer Problem for Neutral Functional Differential Inclusions 227
M. Kisielewicz (Poland)

Smooth Solutions of Linear Equations with Scalar Coefficients in a Right Invertible Operator 232
D. Przeworska-Rolewicz (Poland)

V. APPLICATIONS

Production-Pollution Cycles 263
G. Feichtinger (Austria)

Location Problem and its Applications in Distribution of Petrol Products 277
B. Mażbic-Kulma, E. Komorowska and J. Stępień (Poland)

I. Algorithms and Optimization Methods

QUANTITATIVE STABILITY OF (MIXED-) INTEGER LINEAR OPTIMIZATION PROBLEMS

B. Bank and R. Mandel
Sektion Mathematik
Humboldt Universität Berlin
Berlin, DDR

Abstract.

We treat (mixed-) integer linear optimization problems with variable right-hand sides and an integer constraint matrix from a point of view of quantitative stability, i. e. we are looking for a (quasi-) Lipschitzian characterization of the constraint set-mapping, the value function and the optimal set-function.

Key words: (Mixed-) Integer Programming, Sensitivity, Stability, Parametric Programming, Penalty Functions.

1. Introduction

This paper deals with a stability analysis of the (mixed-) integer linear optimization problem with perturbed right hand sides

$$P(b) \qquad \inf \{c^T x \mid x \in G(b)\}, \qquad b \in R^m$$

where

$$G(b) \underset{Df}{=} \{x \in M(b) \mid x_1, \ldots, x_s \text{ integer}\},$$

$$M(b) \underset{Df}{=} \{x \in R^n \mid Ax = b,\ x \geq 0\},$$

s an arbitrarily fixed integer with $0 \leq s \leq n$.

We suppose throughout all the text that the assumption

(GA) A is an (m,n)-integer matrix

is satisfied.

The aim of our considerations consists in the presentation of quantitative stability results for parametric problems $P(b)$, where quantitative stability will be characterized in terms of the quasi-Lipschitzian behaviour of the constraint sets, the value function

and the optimal set-function.

A multifunction $\Gamma: \Lambda \longrightarrow 2^{R^n}$ $(\Lambda \subset R^m)$ is said to be <u>quasi-Lipschitzian on a subset $\Lambda_o \subset \Lambda$</u> if there are constant numbers ς and τ (not depending on $\lambda \in \Lambda$) such that

$$d(\Gamma(\lambda), \Gamma(\lambda')) \leq \varsigma \| \lambda - \lambda' \| + \tau \qquad \forall \lambda, \lambda' \in \Lambda_o,$$

where $\| \cdot \|$ denotes the euclidean norm and

$$d(C,D) \underset{Df}{=} \inf \{ \varepsilon \mid C \subset U_\varepsilon D,\ D \subset U_\varepsilon C \},$$

the Hausdorff-distance of the sets C and D ($U_\varepsilon C$ is the usual ε-neighbourhood of the set C).

The <u>constraint set mapping G</u>: $B \longrightarrow 2^{R^n}$ is defined by $G(b)$ on $B \underset{Df}{=} \{ b \in R^m \mid G(b) \} \neq \emptyset$.

The <u>value function φ</u> of $P(b)$ defined on B, is given by $\varphi(b) \underset{Df}{=} \inf \{ c^T x \mid x \in G(b) \}$ and its <u>optimal set-function ψ</u> given by the optimal sets $\psi(b) \underset{Df}{=} \{ x \in G(b) \mid c^T x = \varphi(b) \}$.

The theorems presented here extend some of the important results (saying that the mappings G, φ and ψ are quasi-Lipschitzian on B) obtained by BLAIR/JEROSLOW (1977, 1979, 1982) which are derived by the authors in relation to the unified approach in their monography on (nonlinear) integer parametric programming (BANK/MANDEL 1987; see also BANK/BELOUSOV et al. 1986). The constants appearing in this paper with respect to the statements on the quasi-Lipschitzian behaviour of the three mappings defined above are developed on the basis of estimates given there. Another extension of BLAIR's and JEROSLOW's results one can find in a recent paper of COOK et al. (1986) where good estimates for the constants arising in BLAIR/JEROSLOW (1977, 1979, 1982) are shown.

The class of parametric problems $P(b)$ we treat is of interest in its own rights and, similar as in linear programming, important for dealing with duality and exact penalty approaches for (mixed-) integer linear programming.

Applying the presented stability results we prove two theorems, originally due to BLAIR/JEROSLOW (1981), on exact penalty functions for the both cases of pure- and mixed-integer linear optimization.

For convenience we use the assumption that $\varphi(0) = 0$ which ensures that the optimal set $\psi(b) \neq \emptyset$ for all $b \in \operatorname{dom} G = B$; this guarantees that the optimal set $\psi_o(b)$, $b \in B$, corresponding to the linear

programming relaxation $P_o(b)$ of the problem $P(b)$ is non empty, too.

2. Stability Results

In KUMMER (1977) it is shown that the multifunction $C: \operatorname{dom} M \longrightarrow 2^{R^n}$ defined by the compact convex polyhedra

$$C(b) \underset{Df}{=} \{x \in M(b) \mid -e^T x \geq \alpha(b)\} \tag{1}$$

where

$e^T = (1,\ldots,1) \in R^n$,

$\alpha(b) \underset{Df}{=} \min_{1 \leq i \leq r} \{(b^T,-b^T,0^T)(w^i - v^i)\}$,

$\{(v^i,w^i) \mid i=1,\ldots,r\}$ - the set of all vertices of the polyhedron

$W \underset{Df}{=} \{(v,w) \mid (A^T,-A^T,-I)(v-w) = e,\ v,w \geq 0\}$,

I - the (n,n)-identity matrix

is continuous on $\operatorname{dom} M$ and, it holds that

$$M(b) = C(b) + V , \quad b \in \operatorname{dom} M$$

where V is the common recession cone of all $M(b)$, i. e.

$$V \underset{Df}{=} \{x \in R^n \mid Ax = 0,\ x \geq 0\} .$$

Thereby continuity is used in the following sense.
A multifunction $\Gamma: \Lambda \longrightarrow 2^{R^n}$ is said to be <u>upper semicontinuous</u> (according to HAUSDORFF) at $\lambda^o \in \Lambda$ if

$$\forall \varepsilon > 0 \ \exists \delta = \delta(\varepsilon) > 0: \Gamma(\lambda) \subset U_\varepsilon \Gamma(\lambda^o) \quad \forall \lambda \in U_\delta\{\lambda^o\}$$

and, <u>lower semicontinuous</u> (according to BERGE) at $\lambda^o \in \Lambda$ if

$$\forall \Omega \subset R^n \text{ open: } \Omega \cap \Gamma(\lambda^o) \neq \emptyset$$
$$\Longrightarrow \exists \delta = \delta(\Omega) > 0: \Gamma(\lambda) \cap \Omega \neq \emptyset \quad \forall \lambda \in U_\delta\{\lambda^o\} .$$

Γ is said to be <u>continuous</u> at $\lambda^o \in \Lambda$ if it is both upper and lower semicontinuous.

For the compact set $C(b)$ from (1) there is a constant number ϱ such that

$$\|x\| \leq \varrho \|b\| \qquad \forall x \in C(b) , \tag{2}$$

which one can prove using CRAMER's rule and HADAMARD's inequality (see e. g. BANK/MANDEL 1987). Then one easily verifies that C, defined by (1), is Lipschitzian on dom M, i. e. there is a constant ϱ such that

$$d(C(b),C(b')) \leq \varrho \|b - b'\| \qquad \forall\, b,b' \in \operatorname{dom} M \tag{3}$$

which is a version of HOFFMAN's (1952) famous result (see e. g. WALKUP/WETS (1969)). A trivial consequence of these results is the fact that the constraint set mapping M of the linear programming relaxation $P_o(b)$ of the integer problem P(b) shows continuity on dom M. Furthermore M is Lipschitzian on dom M with the same constant number ϱ appearing in (3).

One of our goals is to establish comparable propositions on the constraint set mapping G of the problem P(b) and, in particular, for its optimal set function ψ.
In BANK (1977) the following sets

$$\begin{aligned} K(b) = \{x \in R^n \mid x = u + v,\ u \in C(b),\ v = \sum_{i=1}^{r} \alpha_i v^i, \\ \alpha_i \in [0,1],\ i = 1,\dots,r, \\ x_1,\dots,x_s \text{ integer}\}, \end{aligned} \tag{4}$$

$$V_s \underset{Df}{=} \{x \in V \mid x_1,\dots,x_s \text{ integer}\}$$

are defined where $\{v^1,\dots,v^r\}$ is a system of (mixed-) integer generators of the polyhedral cone V (such systems exists by (GA) and, then $K(b) \neq \emptyset$ on B) and C(b) is taken from (1). The multifunction $K\colon B \longrightarrow 2^{R^n}$ is compact-valued, upper semicontinuous on B and, it holds that

$$G(b) = K(b) + V_s \qquad \forall\, b \in B\ . \tag{5}$$

((GA) is an essential assumption for the upper semicontinuity of K.)
Consequences of these facts are that G is upper semicontinuous on B and that the set B is closed.
Setting for e. g. $\tau = \sum_{i=1}^{r} \|v^i\|$ and applying the estimate (2) one obtains that

$$\|x\| \leq \varrho \|b\| + \tau \qquad \forall\, x \in K(b)\ . \tag{6}$$

In order to construct a compact-valued multifunction $K^*\colon B \longrightarrow 2^{R^n}$ having the property (5) and which additionally shows quasi-Lipschitzian

behaviour on B we make use of the uniform distribution of the set G(b) in the polyhedron M(b). We say that the set
$T_s \underset{Df}{=} \{x \in T \mid x_1, \ldots, x_s \text{ integer}\}$ is uniformly distributed in $T \subset R^n$ if $T_s \neq \emptyset$ and $\delta = \sup_{x \in T} \inf_{y \in T_s} \|x-y\| < \infty$, i. e. there is a δ such that

$$U_\delta(x) \cap T_s \neq \emptyset \qquad \forall x \in T$$

(see e. g. BELOUSOV (1977)).
If (GA) is satisfied, then one easily shows that $G(b) \neq \emptyset$ is uniformly distributed in M(b); in Theorem 1. it will also be shown that the corresponding $\delta(b)$ is bounded from above on the set B.

Theorem 1.
Let (GA) be satisfied. Then

(i) G is quasi-Lipschitzian on B;

(ii) there is a constant number $\delta > 0$ (not depending on b) such that

$$\delta \geq \sup_{x \in M(b)} \inf_{y \in G(b)} \| x-y \| \qquad \forall b \in B;$$

(iii) there is a compact-valued, quasi-Lipschitzian multifunction $K^*: B \longrightarrow 2^{R^n}$ such that

$$G(b) = K^*(b) + V_s \qquad \forall b \in B.$$

Proof: (i) We define the sets

$$\tilde{G}(q) \underset{Df}{=} \{(y,u) \in R^{2n} \mid Ay - Au = q,\ y \geq 0,\ u \geq 0,\ y_1, \ldots, y_s,\ u_1, \ldots, u_s \text{ integer}\}$$

and (7)

$$\tilde{V}_{2s} \underset{Df}{=} \{(y,u) \in R^{2n} \mid Ay - Au = 0,\ y \geq 0,\ u \geq 0,\ y_1, \ldots, y_s,\ u_1, \ldots, u_s \text{ integer}\}.$$

Then, defining with respect to $\tilde{G}(q)$ compact sets $\tilde{K}(q)$ according to (4) and using constant numbers ς and τ such that (6) holds for $\tilde{K}(q)$ one has

$$\tilde{G}(q) = \tilde{K}(q) + \tilde{V}_{2s} \qquad \forall q \in \operatorname{dom} \tilde{G}$$

and (8)

$$\|(y,u)\| \leq \varsigma \|q\| + \tau \qquad \forall (y,u) \in \tilde{K}(q).$$

For arbitrarily fixed points $b, b^* \in B$ and $x \in G(b)$ we construct a

point $x^* \in G(b^*)$ such that

$$\|x - x^*\| \leq 2\varrho\|b - b^*\| + 2\tau \ . \tag{9}$$

Let $\bar{x} \in G(b^*)$ be fixed. Then, $(x,\bar{x}) \in \tilde{G}(b-b^*)$ and, by (8), there are points $(y,u) \in \tilde{K}(b-b^*)$ and $(\hat{y},\hat{u}) \in \tilde{V}_{2s}$ such that

$$(x,\bar{x}) = (y,u) + (\hat{y},\hat{u}) \ .$$

The point $x^* \underset{Df}{=} u + \hat{y}$ belongs to $G(b^*)$ and satisfies (9). Now the validity of (i) is an easy conclusion.

In order to verify the proposition (ii) we set

$$\hat{M} \underset{Df}{=} \{(x,b) \in R^{n+m} \mid Ax - b = 0, \ x \geq 0\}$$

and

$$\hat{G} \underset{Df}{=} \{(x,b) \in \hat{M} \mid x_1,\ldots,x_s \text{ integer}\} \ .$$

Obviously, the set $\hat{G}$ is uniformly distributed in $\hat{M}$. We assume

$$\alpha > \sup_{x\in\hat{M}} \inf_{y\in\hat{G}} \|x - y\| \ .$$

If $x \in M(b)$, then $(x,b) \in \hat{M}$ and there is a point $(x^o,b^o) \in \hat{G}$ with

$$\|(x,b) - (x^o,b^o)\| < \alpha \ .$$

Since $x^o \in G(b^o)$ there is by (i) a point $x^* \in G(b)$ such that (9) holds, for these points. Now one can estimate

$$\|x - x^*\| \leq \|x - x^o\| + \|x^o - x^*\| < \alpha + 2\varrho\|b^o - b\| + 2\tau$$

$$< \alpha + 2\varrho\alpha + 2\tau \underset{Df}{=} \delta \ .$$

(iii) Let $\{v^1,\ldots,v^r\}$ be a system of (mixed-) integer generators of V and, let $\tau > \max\{\delta, \sum_{i=1}^{r} \|v^i\|\}$ be where δ is a constant with the property (ii). Using $C(b)$ from (1) we define

$$K^*(b) \underset{Df}{=} \{x \in R^n \mid x \in G(b), \ x = y+v, \ y \in C(b), \ \|v\| \leq \tau\}. \tag{10}$$

Obviously, $K^*(b)$ is compact and it holds that $G(b) = K^*(b) + V_s$, $b \in B$. To show the quasi-Lipschitzian behaviour of $K^*\colon B \to 2^{R^n}$ let $b,b^* \in B$ and $x \in K^*(b)$ be arbitrarily fixed. Then there are points $y \in C(b)$ and v with $\|v\| \leq \tau$ such that $x = y + v$. From (3) we can conclude that there is a point $y^* \in C(b^*)$ such that

$$\|y - y^*\| \leq \varrho\|b - b^*\| \ .$$

By $C(b^*) \subset M(b^*)$ and proposition (ii) there is a point $x^* \in G(b^*)$ with $\|x^* - y^*\| < \delta$. Now one can estimate

$$\|x - x^*\| = \|y + v - y^* + y^* - x^*\| \leq \|y - y^*\| + \|v\| + \|y^* - x^*\|$$
$$\leq \varrho\|b - b^*\| + \tau + \delta ,$$

and the quasi-Lipschitzian behaviour of K^* is shown. #

The essential result for a quantitative stability analysis of P(b) consists in the following theorem.

Theorem 2.
Let (GA) be satisfied and let $\varphi(0) = 0$. Then

(i) the optimal set-function ψ is quasi-Lipschitzian on B, moreover there is a quasi-Lipschitzian, compact-valued multifunction $\hat{K}: B \longrightarrow 2^{R^n}$ such that

$$\psi(b) = \hat{K}(b) + \hat{V}_s , \qquad b \in B$$

where

$$\hat{V}_s \underset{Df}{=} \{x \in R^n \mid Ax = 0, \quad c^T x \leq 0, \; x \geq 0, \; x_1, \ldots, x_s \text{ integer}\};$$

(ii) the value function is quasi-Lipschitzian on B.

Proof: We show the first part of (i) and using this we verify (ii). Exploiting both the second part of (i) can be shown.
By assumption, $\psi(b) \neq \emptyset$ for all $b \in B$. For $b, b^* \in B$, $x \in \psi(b)$ and $x^* \in \psi(b^*)$, (all this points arbitrarily chosen) and, using $\tilde{G}(q)$ and $\tilde{V}_{2s}$ as defined in (7) with a $\tilde{K}: \text{dom } \tilde{G} \longrightarrow 2^{R^n}$ such that (8) holds, we have $(x, x^*) \in \tilde{G}(b - b^*)$. Further, there are points $(y,u) \in \tilde{K}(b-b^*)$ and $(\bar{y},\bar{u}) \in \tilde{V}_{2s}$ such that

$$(x,x^*) = (y,u) + (\bar{y},\bar{u}) ,$$
$$\|(y,u)\| \leq \varrho\|b - b^*\| + \tau .$$

One easily shows that $\bar{y} \in \psi(A\bar{y})$ and $\bar{u} \in \psi(A\bar{u})$. Since $A\bar{y} = A\bar{u}$ one has $c^T\bar{y} = c^T\bar{u}$ and, for the point $\bar{x} \underset{Df}{=} u + \bar{y}$

$$c^T\bar{x} = c^T u + c^T\bar{y} = c^T u + c^T\bar{u} = c^T x^*$$

holds from which $\bar{x} \in \psi(b^*)$ follows. Now the first part of (i) is an immediate consequence of the estimation

$$\|x - \bar{x}\| = \|y + \bar{y} - u - \bar{y}\| = \|y - u\| \leq 2\|(y,u)\| \leq 2\varrho\|b - b^*\| + 2\tau .$$

If b, b^*, x and x^* have the same meaning as before then

$$|\varphi(b) - \varphi(b^*)| = |c^T(x - x^*)| \leq \|c\| 2\rho \|b - b^*\| + 2\tau \qquad (11)$$

is valid which implies (ii).
It remains to show the second part of (i). Let

$$N(b,a) \underset{Df}{=} \{x \in G(b) \mid c^T x \leq a\}$$

and

$$\hat{V} \underset{Df}{=} \{x \in V \mid c^T x \leq 0\} \ .$$

By (GA) the polyhedral cone V has a system of integer generators. Since $\varphi(0) = 0$, it follows that $c^T v \geq 0$ on V, saying that a subsystem of an integer generating system for V generates $\hat{V}$, too. Therefore, $\hat{V}$ has an integer generating system (independent of the fixed $c \in R^n$). On this basis following a similar way as used in the proof of Theorem 1. one can construct a quasi-Lipschitzian, compact-valued multifunction $\bar{K}$: dom $N \longrightarrow 2^{R^n}$ such that

$$N(b,a) = \bar{K}(b,a) + \hat{V}_s \qquad \forall (b,a) \in \text{dom } N$$

where $\hat{V}_s \underset{Df}{=} \{x \in \hat{V} \mid x_1, \ldots, x_s \text{ integer}\}$.
Let α and β be constants such that

$$d(\bar{K}(b,a), \bar{K}(b^*,a^*)) \leq \alpha \|(b,a) - (b^*,a^*)\| + \beta$$

for all (b,a), $(b^*,a^*) \in$ dom N. Then for

$$\hat{K}(b) \underset{Df}{=} \bar{K}(b,\varphi(b))$$

one has

$$\psi(b) = \hat{K}(b) + \hat{V}_s \qquad \forall b \in B.$$

Applying (ii) and taking (11) into account we get

$$\begin{aligned} d(\hat{K}(b),\hat{K}(b^*)) &\leq \alpha \|(b,\varphi(b)) - (b^*,\varphi(b^*))\| + \beta \\ &\leq \alpha(\|b - b^*\| + |\varphi(b) - \varphi(b^*)|) + \beta \\ &\leq \alpha(\|b - b^*\| + 2\|c\| \varrho \|b - b^*\| + 2\tau) + \beta \\ &= \alpha(1 + 2\|c\| \varrho)\|b - b^*\| + 2\tau\alpha + \beta \end{aligned}$$

which completes the proof. #

We remark that there is only a finite number of different $\hat{V}$ possible to describe $N(b,a)$ if c is allowed to vary.

The next theorem is devoted to relations between the (mixed-) integer problems $P(b)$ and their linear programming relaxations $P_o(b)$ (i. e. the case $s = 0$; by $\varphi_o(b)$ and $\psi_o(b)$ we denote the value function and the optimal set-function of $P_o(b)$, respectively).

As usual in convex analysis we use the notation of the epigraph of a real-valued function f on R^n, i. e.

$$\text{epi } f = \{(x, x_{n+1}) \in R^{n+1} \mid x_{n+1} \geq f(x),\ x \in \text{dom } f\}.$$

Theorem 3.
Let (GA) be satisfied and let $\varphi(0) = 0$. Then

(i) there is a constant Δ such that

$$d(\psi(b), \psi_0(b)) \leq \Delta \qquad \forall\, b \in B;$$

(ii) $\sup \{\varphi(b) - \varphi_0(b) \mid b \in B\} < \infty$;

(iii) the epigraph of the value function φ_0 coincides with the convex hull of the epigraph of the value function φ.

Proof. Exploiting Theorem 2. one obtains the propositions (i) and (ii). Applying linear parametric programming one shows (iii) (see e. g. Nožička et al. (1974)). #

In the proof of the next theorem we use the following decomposition of $x \in R^n$ into subvectors $z = (x_1, \ldots, x_s)^T \in R^s$ and $y = (x_{s+1}, \ldots, x_n)^T$; the matrix A and the vector c are decomposed into $A = (A_1, A_2)$ and $p = (c_1, \ldots, c_s)^T$, $q = (c_{s+1}, \ldots, c_n)^T$, respectively.

Theorem 4.
Let (GA) be satisfied and let $\varphi(0) = 0$. Further, let $b^* \in B$ be any fixed parameter. Then there is a number $\vartheta = \vartheta(b^*) > 0$ such that

$$\varphi(b) \geq \varphi(b^*) - \vartheta \|b - b^*\| \qquad \forall\, b \in B.$$

Proof. Defining sets $K(b)$ and V_s according to (4) we have that (5) holds and, that $K: B \longrightarrow 2^{R^n}$ is compact-valued and upper semi-continiuous. From the proof of the second part of the proposition (i) in Theorem 2. we know that $\psi(b) \cap K(b) \neq \emptyset$, for all $b \in B$. Now let $\{z^1, \ldots, z^r\}$ be the finite set of all different integer vectors $z^i \in R^s$ for which there are vectors $y \in R^{n-s}$ such that $x = (z^i, y) \in K(b^*)$. Let $\epsilon > 0$ be sufficiently small such that the implication

$$x = (z, y) \in U_\epsilon K(b^*) \implies z \in \{z^1, \ldots, z^r\}$$

holds. Then the uppersemicontinuity of K implies the existence of a number $\delta = \delta(b^*) > 0$ with

$$K(b) \subset U_\epsilon K(b^*) \qquad \forall\, b \in U_\delta(b^*)$$

and, one concludes that

$$b \in U_\varepsilon(b^*),\ (z,y) \in K(b) \Longrightarrow z \in \{z^1,\ldots,z^r\}.$$

Using the definitions

$$\tilde{\varphi}(\lambda) \underset{Df}{=} \inf\{q^T y \mid A_2 y = \lambda,\ y \geq 0\},\quad \lambda \in \operatorname{dom}\tilde{\varphi}$$

$$I(b) \underset{Df}{=} \{i \in \{1,\ldots,r\} \mid \exists (z^i,y) \in K(b)\},$$

we can write, for all $b \in U_\delta(b^*)$,

$$\varphi(b) = \inf_{i \in I(b)} \{p^T z^i + \tilde{\varphi}(b - A_1 z^i)\}$$

where $I(b^*) = \{1,\ldots,r\}$.

By Theorem 2. (ii) (the case $s = 0$) there is a constant number $\gamma_1 > 0$ such that

$$|\tilde{\varphi}(\lambda) - \tilde{\varphi}(b^* - A_1 z^i)| \leq \gamma_1 \|\lambda - b^* + A_1 z^i\|,$$

for all λ for which there is a point $y \geq 0$ with $A_2 y = \lambda$ and all $i \in \{1,\ldots,r\}$. From the obvious inequality

$$\tilde{\varphi}(\lambda) \geq \tilde{\varphi}(b^* - A_1 z^i) - |\tilde{\varphi}(\lambda) - \tilde{\varphi}(b^* - A_1 z^i)|$$

one obtains the estimation

$$\tilde{\varphi}(\lambda) \geq \tilde{\varphi}(b^* - A_1 z^i) - \gamma_1 \|\lambda - b^* + A_1 z^i\|, \tag{12}$$

for all $\lambda \in \operatorname{dom}\tilde{\varphi}$ and $i \in \{1,\ldots,r\}$.

If now $b \in U_\delta(b^*)$, then there is an index $i_o \in I(b)$ such that $\varphi(b) = p^T z^{i_o} + \tilde{\varphi}(b - A_1 z^{i_o})$ which yields by (12) the estimate

$$\varphi(b) \geq p^T z^{i_o} + \tilde{\varphi}(b^* - A_1 z^{i_o}) - \gamma_1 \|b - A_1 z^{i_o} - b^* + A_1 z^{i_o}\|$$

$$\geq \varphi(b^*) - \gamma_1 \|b - b^*\|,$$

for all $b \in U_\delta(b^*)$.

On the other hand, if $b \notin U_\delta(b^*)$ then, using constants γ, η such that

$$|\varphi(b) - \varphi(b^*)| \leq \gamma \|b - b^*\| + \eta \qquad \forall\, b, b^* \in B$$

(which exist by Theorem 2. (ii)), one obtains

$$\varphi(b) \geq \varphi(b^*) - |\varphi(b) - \varphi(b^*)| \geq \varphi(b^*) - (\gamma + \frac{\eta}{\delta})\|b - b^*\|.$$

Setting $\vartheta = \vartheta(b^*) = \max\{\gamma_1, \gamma + \eta\delta^{-1}\}$ the proof is completed. #

3. Applications to Exact Penalty Functions for Pure- and Mixed-Integer Linear Optimization

The behaviour of penalty methods essentially depends on the involved penalty functions. In integer programming a particular difficulty appears in finding an appropriate penalty term. Using the results of the previous section we point out that there exist exact penalty functions for pure- and mixed-integer linear optimization problems with a structure allowing the design of solution methods.
Let $P(b)$ be defined as before. For a fixed right-hand side $b^* \in B$ of the linear constraints and a number $r \geq 0$ we associate with $P(b^*)$ the mixed-integer problem

$$P^* \quad \inf \{ c^T x + r\|b^* - Ax\| \mid x \geq 0,\ x_1, \ldots, x_s \text{ integer} \}.$$

By φ^* and ψ^* we denote the corresponding optimal value and the optimal set, respectively.

Theorem 5.
Let (GA) be satisfied and, let $\varphi(0) = 0$. If $b^* \in B$ is fixed and, if r is a sufficiently large number, then

$$\psi(b^*) = \psi^* \quad \text{and} \quad \varphi(b^*) = \varphi^*$$

hold.

Proof. From Theorem 4. we can conclude that

$$\varphi(b) \geq \varphi(b^*) - \vartheta \|b - b^*\| \qquad \forall\, b \in R^m$$

where $\varphi(b) = +\infty$ if $b \notin B$. For $r > \vartheta$ and $x \geq 0$, $x_1, \ldots, x_s$ integer such that $Ax \neq b^*$ we obtain the estimate

$$\begin{aligned} c^T x + r\|b^* - Ax\| &\geq \varphi(Ax) + r\|b^* - Ax\| \\ &> \varphi(Ax) + \vartheta \|b^* - Ax\| \geq \varphi(b^*) \end{aligned}$$

from which the proposition is an immediate consequence. ∎

We consider now the pure-integer problem, (i. e. $s = n$)

$$P(b) \quad \inf \{ c^T x \mid Ax = b,\ x \geq 0,\ x \text{ integer} \}$$

for a fixed $b \in B$. Let

$$L(x,u,r) \underset{Df}{=} (c - u^T A)^T x + r\|Ax - b\|^2 + u^T b.$$

Theorem 6.
Let (GA) be satisfied, $\varphi(0) = 0$ and $b \in B$ a fixed parameter. Then, for any $u \in R^m$ there exists a number $r(u) \geq 0$ (not depending on b)

such that

$$L(x^*,u,r) = \min \{L(x,u,r) \mid x \geq 0,\ x \text{ integer}\}$$

holds if and only if $x^* \in \Psi(b)$, for all $r \geq r(u)$. Moreover,

$$L(x^*,u,r) = \varphi(b) .$$

Proof. On account of $B \subset \{x \in R^n \mid x \text{ integer}\}$, Theorem 2. yields the existence of a constant number $\varrho > 0$ such that

$$|\varphi(b) - \varphi(b')| < \varrho \|b - b'\| \qquad \forall b,b' \in B.$$

Further, $\|b - b'\| \leq \|b - b'\|^2$, for all $b,b' \in B$. For an arbitrarily fixed $u \in R^m$ we put

$$r(u) \underset{Df}{=} 1 + \varrho + \|u\| .$$

Now, let $x \geq 0$, x integer and $Ax \neq b$, and let $r \geq r(u)$. Then

$$\begin{aligned} L(x,u,r) &= c^T x + u^T(b - Ax) + r\|Ax - b\|^2 \geq \\ &\geq \varphi(Ax) + u^T(b - Ax) + r\|Ax - b\| \geq \\ &\geq \varphi(b) - |\varphi(b) - \varphi(Ax)| + u^T(b - Ax) + r\|Ax - b\| \geq \\ &\geq \varphi(b) - \varrho\|b - Ax\| - \|u\|\,\|b - Ax\| + r\|Ax - b\| > \varphi(b), \end{aligned}$$

which the propositions of the theorem are simple consequences of. #

Remarks.

Theorem 6. shows that $L(x,u,r)$ yields a penalty function (for the pure integer problem $P(b)$) which, obviously, possesses "nicer" properties than the one considered in Theorem 5.

Bounds for the "sufficiently large" numbers r occuring in both the Theorems 5. and 6. can be derived using estimates of the constants taken from the Theorems 2. and 4. or of these given by COOK et al. (1986).

References

Bank, B. (1977). Quantitative Stabilitätsuntersuchungen rein- und gemischt-ganzzahliger linearer parametrischer Optimierungsprobleme. Diss. (B), Humboldt-Universität Berlin.

Bank, B., E. G. Belousov, R. Mandel, J. N. Čeremnych and V. M. Shironin (1986). Matematyceskije programmirowanje: voprozy rasreshimosti i ustoyčivosti. Izd. MGU, Moskwa (russian).

Bank, B., R. Mandel (1987). Parametric Integer Optimization. Akademie Verlag Berlin, Mathematical Research Vol. 39.

Belousov, E. G. (1977). Vvedenie v vypuklyj analiz i celočislennoe programmirovanje. Iszd. MGU, Moskwa (russian).

Blair, C. E., R. G. Jeroslow (1977). The value function of a mixed integer program: I. Discrete Mathematics, 19., 121-138.

Blair, C. E., R. G. Jeroslow (1979). The value function of a mixed integer program: II. Discrete Mathematics, 25, 7-19.

Blair, C. E., R. G. Jeroslow (1981). An exact penalty method for mixed integer programs. Mathematics of operations research, 6, No. 1, 14-18.

Blair, C. E., R. G. Jeroslow (1982). The value function of an integer program. Mathematical Programming, 23, 237 - 273.

Cook, W., A. M. H. Gerhards, A. Schrijver, E. Tardos (1986). Sensitivity Theorems in integer linear programming. Mathematical Programming, 34, 251-264.

Hoffmann, A. J. (1952). On approximate solutions of systems of linear inequalities. Journal of Research of the National Bureau of Standards, 49, 263-265.

Kummer, B. (1977). Globale Stabilität quadratischer Optimierungsprobleme. Wiss. Z. der Humboldt Universität Berlin, Math.-Nat. R. XXX, 5, 565-569.

Nožička, F., J. Guddat, H. Hollatz, B. Bank (1974). Theorie der linearen parametrischen Optimierung. Akademie Verlag Berlin.

Walkup, D., R. Wets (1969). A Lipschitzian characterization of convex polyhedra. Proc. Amer. Math. Soc., 23, 167-173.

PARALLEL HYBRID OPTIMIZATION METHODS

by O. Burdakov and C. Richter

Computer Center of the Academy of Sciene Moscow,
USSR-117967 Moscow-W-333, u. Varilova

Abteilung Mathematik/Rechentechnik der
Ingenieurhochschule Köthen,
DDR-4370 Köthen, Bernburger Str. 52 - 57

1. Introduction

Given the unconstrained minimization problem

$$f(x) = \min! \qquad \text{s.t.} \quad x \in R^n \tag{1}$$

and the nonlinear programming problem

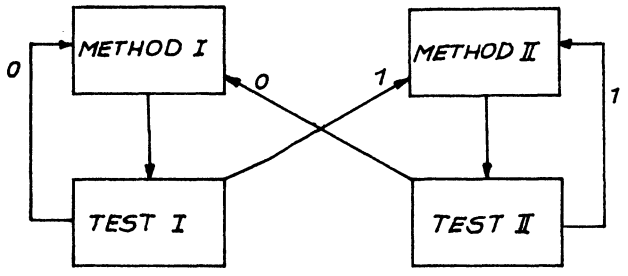
$$f(x) = \min! \qquad \text{s.t.} \quad x \in G \tag{2}$$

where $G = \{ x, : g_j(x) \leq 0 \qquad j=1,\dots,m \}$.

To obtain a globally and locally superlinearly convergent serial method it is possible to combine a globally convergent method I and a locally superlinearly convergent method II for a serial computer.

These hybrid methods can be regarded as realizations of the following general principle:

After having generated the iterates x^k resp. $z^k = (x^k, u^k)$ (an approximation of the KUHN-TUCKER-point $z^* = (x^*, u^*)$ by one step of the method I and II, respectively, we decide by use of the convergence Tests I and II, respectively, whether the generation of x^{k+1} resp. z^{k+1} by method II is advantageous or not.



Examples of this principle are:
for problem P1:

method I: Method of deepest descent

method II: Newtons method

for problem P2:

method I: Penalty-Barrier-method

method II: Wilsons method

The aim of the paper presented is to transform the prescribed principle on parallel computers. The method obtained has the important advantage that the convergence is globally and locally quadratically, indepedet from the dimension of the considered problem. This is a different to the sequential secant method with the trend to linear convergence, if the dimension is increasing. The extension of the prescribed technique on the treatment of problem (2) is obvious.

2. Notations and preliminaries

Denote a set of n+2 points by $\bar{x}^k = [x^{k,1},\ldots,x^{k,n+2}]$.
In the following algorithm when we write, that $\bar{x}^{k+1}$ is computed parallely with the use of some methods, it means that each $x^{k+1,i}$ $(i=1,\ldots,n+2)$ is computed by the i-th processor with the use of this method on the basis of the prints $x^{k,j}$ $j=1,\ldots i-1,i+1,\ldots n+2$.
We indroduce METHOD 1, METHOD 2, TEST 1 and TEST 2 in the following way:

METHOD 1:

Given n+1 points $x^1,\ldots,x^{n+1}$. Choose ε with $0 < \varepsilon < 1$ and $\underline{\varepsilon}$ with $\underline{\varepsilon} \ll \varepsilon$

Set $l = 0$

S1: Set $y^1 = x^1$

$$y^i = x^1 + (\tfrac{1}{2})^l (x^i - x^1) \qquad (i=2,\ldots,n+1)$$

S2: Compute the direction g as a solution of the linear system

$$\begin{bmatrix} y^1 - y^2 \\ \vdots \\ y^1 - y^{n+1} \end{bmatrix} g = \begin{bmatrix} f_1 - f_2 \\ \vdots \\ f_1 - f_{n+1} \end{bmatrix}$$

where $f_i = f(y^i)$ $\quad (i=1,\ldots,n+1)$

S3: Determine the greatest value of $\alpha = (\frac{1}{2})^j$ $\quad (j=1,2,\ldots)$, such that

$$f(y^1 - \alpha g) \leq f(y^1) - \alpha \varepsilon \, \| g \|^2$$

S4: If an α exists, such that $\alpha > \underline{\varepsilon}$, then set

$$x^{n+2} = y^1 - \alpha g$$

else set $l=l+1$ and goto S2.

METHOD 2:

Given n+1 points $x^1,\ldots,x^{n+1}$.

S1: Dtermine $\bar{y}$ as a solution of the linear system

$$G(x-x^{n+1}) + f(x^{n+1}) = 0,$$

where

$$G = \Delta F \circ \Delta X^{-1}$$

with

$$\Delta F = \left[f(x^1) - f(x^2),\ldots, f(x^n) - f(x^{n+1}) \right]$$

and

$$\Delta X = \left[x^1 - x^2,\ldots, x^n - x^{n+1} \right]$$

S2: Set $s = \bar{y} - x^{n+1}$

Determine the greatest value of $\alpha = (\frac{1}{2})^j$ $(j=1,2,\ldots)$ such that

$$f(x^{n+1} + \alpha s) - f(x^{n+1}) \leq \frac{1}{2} \nabla f(x^{n+1})^T s$$

and set

$$y = x^{n+1} + \alpha s$$

Test 1: If $\| g^{k+1,i} \| \leq \underline{\varepsilon}$ for $i \in I^{k+1} \subset \{1,\ldots,n+2\}$ then determine the Indes i_o with:

$$f(x^{k+1,i_o}) = \min_{i \in I^{k+1}} f(x^{k+1,i})$$

set $w^{k+1,i} = x^{k+1,i}$ $\quad (i=1,\ldots,n+2)$.

$x^{k+1,j} = y^{i_o,j}$ $\quad (j=1,\ldots,n+2)$

and METHOD = METHOD 2.

Test 2: If $\|\nabla f(x^{k+1,i})\| \geq c_1 \min_{1 \leq i \leq n+2} f(x^{k,i})$

for all i, then set

$$\underline{\varepsilon} = \underline{\varepsilon} \cdot c_2,$$
$$x^{k+1,i} = w^{k+1,i}$$
and METHOD = METHOD 1.

3. Algorithm

S0: Choose a set of n+2 initial points
$x^{0,1},\ldots,x^{0,n+2}$; $\varepsilon > 0$; $c_1, c_2 \in (0,1)$; $\underline{\varepsilon} > 0$

k-th step of the iteration ($k \geq 0$):

S1: Determine $\bar{x}^{k+1}$ parallely with the use of METHOD.

S2: TEST

S3: Set k=k+1. Goto S1.

4. Properties of METHOD 1 and METHOD 2

For the proof of the global convergence of METHOD 1 we consider the generated sequence $\{w^k\}$ with the initial points

$$w^{-n}, w^{-n+1},\ldots,w^0 \in R^n.$$

At first the following conditions are formulated.

Condition (C1): We say that the function f(x) fulfilles the condition (C1), if it holds:

(i): $\exists \tau$: $f(x) \geq \tau$ $\forall x \in R^n$

(ii): for a given $x^0 \in R^n$: $W(x^0) = \{x: f(x) \leq f(x^0)\}$ is closed

(iii): $f(x)$ is differentiable on $W(x^0)$ and for all $x,y \in W(x^0)$ with the property

$$[x,y] = \{z : z = \lambda x + (1-\lambda)y: \lambda \in 0,1\} \subset W(x^0)$$

the relation

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\| \qquad (L > 0)$$

is satisfied.

Condition (C2): For a given $\omega \geq 0$ we say that n Vektors $d^i = w^k - w^{k-1}$ (i=k-1,...,k-n) are w-nonsingular, if

$$\det \left(\frac{d^1}{\|d^1\|}, \ \ldots \ \frac{d^h}{\|d^n\|} \right) \geq \omega$$

Theorem 1: We assume, that for a given w^0 the function f(x) fulfiles the condition (C1) and that n+1 points w^{k-i} (i=0,...,n) fulfil condition (C2) for all k (independently from k).
Then the sequence $\{w^k\}$ is well-defined for all k and it holds

$$\liminf_{k\to\infty} \|g^k\| = 0 \quad \text{and} \quad \liminf_{k\to\infty} \|\nabla f(w^k)\| = 0.$$

Before we discus properties of METHOD 2 some conditions for the minimum point x^* and the function f(x) are formulated.

Condition (C3): We say that the condition C3 is satisfied, if for the minimum point $x^* \in R^n$ and for some ball $S^* = S(x^*, \sigma)$ it holds:

i) f is twice differentiable on S

ii) $\nabla f(x^*) = 0$

iii) the Hessian $H(x^*)$ is positiv definit

iv) H is Lipschitz-continuous on S^* with a constant L, i.e.

$$\|H(x) - H(y)\| \leq L \ \|x - y\| \qquad \forall x,y \in S^*$$

If C3 is satisfied then the point x^* is a so-called regular minimum-point.

Theorem 2: Let x^* be a regular minimum-point and let $x^{-n},\ldots,x^0$) be a set of points containing in a suitable neighboorhood $S(x^*,\sigma)$ of x^* and satisfying C1.
Then the sequence $\{x^k\}$ generated by METHOD 2 is well defined. Furthermore, either the algorithm terminates after a finite number of iterations at the regular minimum point $x^N = x^*$ of f or x^k is infinite and $\lim_{k\to\infty} x^k = x^*$.
The rate of convergence α is the positive root of. $\alpha^{n+1} = \alpha^n + 1$

Sketch of the proof:

It is possible to prove, that under the required assumptions the generated matrix G is a strong consistent approximation of $H(x^k)$. Therefrom and from the assumption, that the iterates $x^k, x^{k-1}, \ldots, x^{k-1}$ are contained in the ball $S(x^*, \sigma)$ it is possible to derive the nonsingularity of $H(x^k)$ which implies the existence of x^{k+1} and the relation

$$\| x^{k+1} - x^* \| \leq K\Big(\sum_{i=0}^{n} \| x^{k-i} - x^* \|\Big) \| x^k - x^* \| . \qquad (8)$$

If the inequality

$$K\Big(\sum_{i=0}^{n} \| x^{k-i} - x^* \|\Big) < 1$$

holds, then the point x^{k+1} belongs to the ball $S(x^*, \sigma)$. It is clear, that $\lim_{k \to \infty} x^k = x^*$.

5. Convergence Analysis

In order to investigate convergence properties of the iterates $\bar{x}^k$ generated by the hybrid algorithm we define the index set

$$K_1 = \{ k : \bar{x}^{k+1} \text{ is generated by METHOD 1} \}$$

For all remaining iteration indices $k \notin K_1$ the iterates $\bar{x}^{k+1}$ are determined by METHOD 2.
Now we consider the index set K_1 corresponding to METHOD 1.
With the presented theorems we can conclude that K_1 is finite because both TEST 1 and TEST 2 lead to the locally convergent method for every safficienthly large k.
We summarize the behaviour of $\{\bar{x}^k\}$ in the following theorem.

Theorem 3

We assume, that with the points $x^{0,i}$ $(i=1,\ldots,n+2)$ and the level sets $W(x^{0,i})$ the condition (C1) is satisfied.
Furthermore the condition (C2) may be satisfied for every subset of n+1 points $x^{k,i}$ of $\bar{x}^k$ independently from k.

If additionaly each minimum point of f is regular then the generated sequence $\{\bar{x}^k\}$ has the following properties:

i) $\lim_{k\to\infty} x^{k,i} = x^*$

ii) the rate of convergence is at least R - quadratically.

Proof:

The first part of the theorem is obvious from the remarks above.
The second part required considerations on the R-order of sequences $\{x^{k,i}\}$ coupled by using of a parallel algorithm.
If we transform the estimation ( ) on this situation, the system of inequalities

$$\| x^{k+1,i}-x^* \| \leq k_i \sum_{j\neq i} \| x^{k,i}-x^* \| \, \| x^{k,j}-x^* \| \quad (i=1,\ldots,n+2)$$

anses in a natural way.
It is possible to write the relations (8) in the form

$$\| x^{k+1,i}-x^* \| \leq k_i \sum_{j\neq i} \prod_{l=1}^{n+2} \| x^{k,l}-x^* \|^{m^j_{il}}, \qquad (9)$$

where the coefficients m^j_{il} of the matrix M^j are 0 or 1.
For first order difference inequalities of this kind SCHMIDT (/2/) has proved, that the R-order of the underlying sequences of iterates is at least equal to the spectral radias of the matrix of exponents if a positive eigenvector belonging to the spectral radius exists.

For the application of SCHMIDT's result we consider the structur of the matrix M^j. In each row of these matrix there are two elements equal 1, the other elements are equal 0. It is immediatly obvious, that $e = (1,\ldots,1)^T$ is an eigenvector of M. and 2 is the eigenvalue belonging to it.

Remark:
The quadratically rate of convergence diffus from the result in the serial case, where α is the positive root of the equation

$$\alpha^{n+1} = \alpha^n + 1.$$

Depending on the dimension n this root has the following values

n	α
1	1.618030
2	1.465576
4	1.324718
5	1.285191

and it holds

$$\lim_{n \to \infty} \alpha(n) = 1$$

6. Concleeding remarks

It is possible to extence the idea prescribed above to realize a whole class of hybrid methods on parallel computers. Especially using of parallel stochastic search methods as METHOD 1 and derivative-free-methods as METHOD 2 is advantageously.

7. References

/ 1 / KLEINMICHEL, H.; C. RICHTER; K. SCHÖNEFELD:
On the global and superlinear convergence of a discretized version of WILSON's method. Computing 29 (1982) 289 - 307.

/ 2 / SCHMIDT, J. W.: On the R-order of coupled sequences. Computing 26 (1981) 333 - 342.

/ 3 / SCHWETLICK, H.: Numerische Lösung nichtlinearer Gleichungen. VEB Deutscher Verlag der Wissenschaften, Berlin 1979.

Continuous Generalized Gradients for Nonsmooth Functions

V. F. Demyanov

Leningrad State University, Leningrad, USSR

At present there exist many generalizations of the concept of gradient for nonsmooth functions. We mention only those of R. T. Rockafellar [1], F. Clarke [2], J. Warga [3], J.-P. Penot [4], B. Mordukhovich and A. Kruger [5], V. F. Demyanov and A. M. Rubinov [6]. But all these generalizations inherit the main cause of nonsmoothness - the absence of differentiability which gives rise to the discontinuity of corresponding mappings.

In what follows we try to restore the lost continuity (at least for some classes of nonsmooth functions).

Let f be a real-valued function defined on $\mathbb{R}^n$. We say that f is codifferentiable at $x\in\mathbb{R}^n$ if there exist two convex closed sets $\underline{d}f(x)\subset\mathbb{R}^{n+1}$ and $\overline{d}f(x)\subset\mathbb{R}^{n+1}$ such that

$$f(x+\Delta)=f(x)+\max_{[a,v]\in\underline{d}f(x)}[a+(v,\Delta)]+\min_{[b,w]\in\overline{d}f(x)}[b+(w,\Delta)]+o(\Delta,x)$$

where $\frac{o(\alpha\Delta,x)}{\alpha}\xrightarrow[\alpha\downarrow 0]{}0$, for all $\Delta\in\mathbb{R}^n$.

The pair of sets $Df(x)=[\underline{d}f(x),\overline{d}f(x)]$ is called a codifferential of f at x; $\underline{d}f(x)$ is called a hypodifferential and $\overline{d}f(x)$ a hyperdifferential of f at x. It is not difficult to see that the class of codifferentiable functions coincides with the class of quasidiffer-

entiable functions [6]. It is also clear that a codifferential is not uniquely defined.

If there exists a codifferential of f at x of the form $Df(x)=[\underline{d}f(x),\{0_{n+1}\}]$ then we say that the function f is hypodifferentiable at x; and if there exists a codifferential of the form $Df(x)=[\{0_{n+1}\},\bar{d}f(x)]$ then f is called hyperdifferentiable at x.

If f is codifferentiable in a neighbourhood S of x then a codifferential mapping $Df:S \to 2^{\mathbb{R}^{n+1}} \times 2^{\mathbb{R}^{n+1}}$ (where 2^A is the power set of A) is defined. We say that f is continuously codifferentiable at x if f is codifferentiable in a neighbourhood S of x and there exists a codifferential mapping Df which is continuous (in the sense of Haussdorff) at the point x.

Note that all C^1-funtions, all convex and concave functions, all functions of maximum and minimum (over a compact set of C^1 functions) are continuously codifferentiable.

Calculus of codifferentials can be constructed: the set of continuously codifferentiable functions (CC^1-functions) is a linear space closed with respect to all "smooth" operations and (more importantly) those of taking minimum or maximum over a finite number of functions.

Thus, the continuity of generalized gradients is restored by introducing a new concept (that of codifferentiability). We can apply this new concept to construct a new family of numerical methods to optimize CC^1-functions incorporating (with natural adjustments) all existing algorithms for quasidifferentiable optimization (see [7] and [8]).

It is worth mentioning that as it turns out the effectiveness of the "linearization method" of B. N. Pshenichny [9] for minimax

problems is due to the fact that max-functions are continuously hypodifferentiable.

A detailed report will be published elsewhere (see, e. g., [10]).

Remark: The concept of codifferential allows to construct first-order approximations to sets described by codifferentiable functions. This seems to be *extremely promising* from computational standpoint.

References

[1] R. T. Rockafellar. The theory of subgradients and its applications to problems of optimization. Lecture Notes Series, Montreal Univ. Press, Montreal, 1978.

[2] F. H. Clarke. Optimization and nonsmooth analysis. John Wiley & Sons Publ. Co., New York, 1983.

[3] J. Warga. Optimal control of differential and functional equations. Academic Press, New York, 1972.

[4] J.-P. Penot. Calcul sous-différentiel et optimization. J. Funct. Anal. 27, 248-276, 1978.

[5] A. Y. Kruger, B. S. Mordukhovich. Extremal points - the Euler equation in nonsmooth optimization problems. Dokl. Acad. Nauk BSSR 24, 684-687, 1978.

[6] V. F. Demyanov, A. M. Rubinov. Quasidifferential Calculus. Optimization Software, Inc., New York, 1986.

[7] V. F. Demyanov, S. Gamidov, T. I. Sivelina. An algorithm for minimizing a certain class of quasidifferentiable functions. Mathematical Programming Study 29, 74-84, 1986.

[8] D. Pallaschke, P. Recht. On the steepest descent method for a class of quasidifferentiable optimization problems. Lecture Notes in Economics and Math. Systems, Vol. 255, 252-264, 1985.

[9] B. N. Pshenichny. The linearization method. Nauka, Moscow, 1983.

[10] V. F. Demyanov. Codifferentiability and codifferentials of nonsmooth functions (to appear in Doklady of USSR Academy of Sciences).

OUTER CUT METHODS IN GLOBAL OPTIMIZATION

R. Horst
Fachbereich IV - Mathematik
Universität Trier
Postfach 3825
D - 5500 Trier, Fed. Rep. of Germany

Summary. A brief report is presented on recent developments in outer cut methods for solving certain classes of constrained global optimization problems. First, two theorems are stated each of which is helpful to assert convergence of a basic concept. Nonlinear cuts and unbounded feasible sets are admitted. A broad class of new cutting plane methods can be deduced. Moreover, a practically useful constraint dropping strategy is given and, finally, some questions of numerical implementations are discussed including the computation of new vertices generated by a cut and the identification of redundant constraints.

1. Introduction. Cutting plane methods have been used in different fields of optimization for some time. A cutting plane is a linear constraint that is used to reduce a certain set in such a way that does not exclude optimal feasible points of an optimization problem. In the sequel, we will only consider cuts that never cut off parts of the feasible set: outer cut methods. However, nonlinear cuts will be admitted.

We have in mind continuous global optimization problems where nonglobal local minima occur. Let throughout this paper $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a continuous function and let D denote a closed subset of $\mathbb{R}^n$. Furthermore, it is supposed that $\min f(D) := \min \{f(x) : x \in D\}$ exists. We state our problem in the form

$$\text{global } \min_{x \in D} f(x).$$

Integer programming problems,where cut methods also belong to the basic tools,will not be discussed explicitely here. Note however,that integer programming is closely related to certain global optimization problems (cf. e.g. [10], [24]).

Other fields, where cutting plane methods have been extensively investigated, are convex programming and local, possibly nonsmooth optimization.

The first methods applied in convex programming (as well as in certain fields of approximation) were the so-called KCG algorithm (cf.[1],[17]) and the supporting hyperplane approach (cf. [32]). Since however, the numerical behaviour of these methods was not very satisfactory (low convergence rate, stability problems near a solution), several modifications and combinations with other devices were introduced, e.g. line

searches, quadratic approximation etc.. Today, modified cutting plane methods belong to the major tools in nonsmooth optimization (cf. e.g. [8], [9], [18]) and references given there).

Another challenging question with outer cut methods has always been the question of constraint dropping strategies. Since, in the basic version of such a procedure, in each iteration a new constraint is added to the existing set of constraints but no constraint is ever deleted, the size of the problems to be solved increases from iteration to iteration. Therefore,much effort has gone in the development of devices that allow to drop certain previous constraints (e.g. [6],[8],[9],[10],[12],[14], [15], [16], [28]).Of course,the most satisfactory results were obtained for convex and local optimization where the number of constraints to be considered in each iteration is bounded (e.g. [9], [18] and references given there).
However, these nice results concerning speed, stability and constraint dropping obtained in local (or convex) programming could not been carried over to global problems, since, as a rule, local analysis fails in global optimization.
Note as well, that outer cut methods including certain constraint dropping strategies are strongly related to duality and decomposition (e.g. [2], [20] and references given there).

In global optimization, outer cut methods belong to the major tools for solving some basically important classes of problems,the most prominent example being the class of concave minimization problems, where f is a concave function and D is a convex set (cf. e.g. [11],[26]); for recent surveys, see [13], [21], [22]). General theories of outer cut methods that considerably simplify and extend previous theories have been developed recently in [14], [29] and modified and generalized further in [15]. Moreover, new results on certain numerically important questions arising from these methods were recently obtained in [16].
It is the purpose of this paper to briefly summarize these results. Furthermore,some modifications will be given and several open questions for further research will be stated.

In Section 2,the basic concept and two associated theorems will be presented. Section 3 contains a practicable constraint dropping strategy. In Section 4, a general class of cutting plane methods is illustrated and the final Section 5 is devoted to questions of how to solve the subproblems arising in each iteration and how to identify redundant constraints.

2. A Basic Concept

Consider the problem

$$\text{global min}_{x \in D} f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and D is a closed subset of $\mathbb{R}^n$. Suppose that min $f(D)$ exists. Then, an outer cut concept proceeds as follows.

Choose a closed set $D_1 \subset \mathbb{R}^n$ satisfying $D_1 \supset D$. Set $k \leftarrow 1$.

Iteration k (k = 1,2,...):

Solve the "relaxed problem"

$$(P_k) \qquad \text{global min}_{x \in D_k} f(x) \text{ obtaining a solution } x^k. \tag{2}$$

If $x^k \in D$, then stop: x^k solves problem (1). Otherwise construct a constraint function $l_k : \mathbb{R}^n \to \mathbb{R}$ satisfying

$$l_k(x) \leq 0 \quad \forall x \in D, \tag{3}$$

$$l_k(x^k) > 0. \tag{4}$$

Set $D_{k+1} = D_k \cap \{x \in \mathbb{R}^n : l_k(x) \leq 0\}$, $k \leftarrow k+1$ and go to the next iteration.

It is supposed of course, that the subproblems (P_k) can be reasonably efficiently solved (cf. Section 5). The set

$$H_k := \{x \in \mathbb{R}^n : l_k(x) = 0\} \tag{5}$$

strictly separates x^k from D and is called a "cut". In most applications D_1 is a polyhedral set and we have

$$l_k(x) = (a^k, x) + b_k, \tag{6}$$

where $a^k \in \mathbb{R}^n$, $b_k \in \mathbb{R}$ and $(\cdot\,,\,\cdot)$ denotes the inner product in $\mathbb{R}^n$. In this case, H_k is a hyperplane and the method is said to be a cutting plane algorithm.

Two recent results on the convergence of such an outer-cut procedure will be presented below. The first result deals with cutting plane methods and compact sets D_k.

Let $H_k = \{x : (a^k, x) + b_k\}$, $k \in \mathbb{N}$, define a sequence of hyperplanes. Then, $\{H_k\}$ is said to converge to a hyperplane

$H = \{x : (a,x) + b\}$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$ (notation: $H_k \underset{k}{\longrightarrow} H$) if we have

$a^k \underset{k\to\infty}{\longrightarrow} a$, $b^k \underset{k\to\infty}{\longrightarrow} b$.

Theorem 1. (Tuy [29]). Within a cutting plane procedure let D_1 be compact and suppose that for each convergent subsequence $\{x^q\}$ of $\{x^k\}$ satisfying $x^q \underset{q}{\longrightarrow} \bar{x} \notin D$, $H \underset{q}{\longrightarrow} H$, we have that H strictly separates $\bar{x}$ from D. Then, every accumulation point of $\{x^k\}$ solves (1).

The following result generalizes Theorem 1 and a theorem presented by Horst, Thoai and Tuy [14], where a uniform Lipschitzian property of the cut functions l_k was required. Moreover, unbounded feasible sets D and nonlinear cuts are admitted.

Theorem 2. (Horst, Thoai, Tuy [15]). Within the outer cut concept presented above assume that

(i) l_k is continuous for each $k = 1,2,\ldots$ and for each convergent subsequence $\{x^q\} \subset \{x^k\}$ we have $l_q \underset{q}{\longrightarrow} l$ where $l : \mathbb{R}^n \to \mathbb{R}$ is continuous;

(ii) for each convergent subsequence $\{x^q\} \subset \{x^k\}$ satisfying $x^q \underset{q}{\longrightarrow} \bar{x}$, $l_q(x^q) \underset{q}{\longrightarrow} l(\bar{x}) = 0$ we have $\bar{x} \in D$.

Then every accumulation point of the sequence $\{x^k\}$ solves (1).

In contrast to previous rather involved theories (e.g. [6], [12], [14]), Theorem 2 seems to reduce the matter to a basic and simple result. It has been shown in [15] that a broad class of new cutting plane algorithms can be derived that includes known procedures as special cases. Moreover, several constraint dropping strategies can be deduced. The proof of Theorem 2 is very easy:
Let $x^q \underset{q}{\rightarrow} \bar{x}$. Then, by (3) and (4) we see that $l_q(x^{q+1}) \leq 0, l_q(x^q) > 0 \ \forall q$, hence, for $q \to \infty$, we must have $l(\bar{x}) = 0$. From (ii) it follows then that $\bar{x} \in D$. Finally, it is easily seen that, by $f(x^k) \leq \min f(D)$ and by continuity of f, $\bar{x} \in D$ implies $f(\bar{x}) = \min f(D)$.

3. Constraint Dropping

Let

$$H_k^- := \{x \in \mathbb{R}^n : l_k(x) \leq 0\}. \tag{7}$$

Then, in the basic concept described in Section 2, we have

$$D_{k+1} = D_1 \cap \left(\bigcap_{i=1}^{k} H_i \right).$$

We try to drop certain constraints defining D_{k+1}, i.e. we try to find a set of indices $I_k \subset \{1,2,\ldots,k\}$ such that

$$D_{k+1} = D_1 \cap \left(\bigcap_{i \in I_k} H_i \right) \tag{8}$$

defines a convergent outer cut procedure.

In their outstanding paper [6], Eaves and Zangwill presented a general theory of cut procedures and constraint dropping strategies. Cuts are defined via a set valued mapping called a cut map that has to satisfy certain assumptions concerning regularity and separation. Separation is defined by a real-valued function called a separator that, in a very general sense,describes a distance from its argument to the set we want to attain to by the cut procedure.

More recently, it was shown by Horst, Thoai and Tuy [14] that, in the practically most interesting case of cutting planes,the cut map/separator theory can be considerably simplified: new proofs of basic results are given and several classes of constraint dropping strategies are proposed that are of interest in global optimization.

In a later paper [15], it could be shown by the same authors that all these constraint dropping strategies remain even valid if nonlinear cuts are applied. Moreover,it it demonstrated, that the earlier and some new results on outer cut methods can be obtained by a very straightforward access via Theorem 2 that is considerably less involved than the cut map / separator theories.

Below, we present as an example, that subclass of constraint dropping strategies included in the development given in [15] that after first preliminary numerical experiences seems to be promising for practical use.

Let $p : D_1 \longrightarrow \mathbb{R}$ be a continuous function satisfying

$$p(x) > 0 \quad \forall x \in D_1 \setminus D \ , \ p(x) = 0 \ \forall x \in D. \tag{9}$$

$p(x)$ is frequently called a penalty function.

Many penalty functions are known in nonlinear programming. Let, e.g.,

$$D = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \ (i = 1,\ldots,m)\},$$

where $g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ $(i = 1,\ldots,)$ are continuous on D_1. Then, it is well known that all functions of the form

$$p(x) = \sum_{i=1}^{m} [\max \{0, g_i(x)\}]^{\beta}, \ \beta \geq 1 \tag{10}$$

are penalty functions.

Furthermore, let $\{\alpha_i\} \subset \mathbb{R}_+$ be an arbitrary monotonically decreasing sequence satisfying $\alpha_i \underset{i\to\infty}{\longrightarrow} 0$ (notation: $\alpha_i \searrow 0$) and let

$$\varepsilon_{ij} := \alpha_i - \alpha_j \ , \ i \leq j. \tag{11}$$

Take, for example, $\alpha_i = (\alpha)^i$ where $\alpha \in]0,1[$.

Theorem 3. Let $p(x)$ be a penalty function as defined in (9) and let $\{\varepsilon_{ij}\}$ be a double-indexed sequence as defined in (11). Assume that in the outer cut concept the sequence of cut functions satisfies the separation requirements (3) and (4) and the assumptions (i), (ii) of Theorem 2. Then

$$D_{k+1} = D_1 \cap \left(\bigcap_{i \in I_k} H_i^- \right) \quad k = 1,2,... \tag{12}$$

with

$$I_k = \{i : 1 \leq i \leq k,\ p(x^i) \geq \varepsilon_{ik}\} \tag{13}$$

defines a convergent outer cut procedure, i.e., every accumulation point of $\{x^k\}$ solves problem (1).

Remarks. Note that, by $\varepsilon_{kk} = 0$, we always have $k \in I_k$. Moreover, since for fixed i, ε_{ik} is monotonically increasing in k, we have that $i \notin I_k$ implies $i \notin I_s$ $\forall s > k$: a dropped cut remains dropped forever.
For the same reason, a cut retained at some iteration may be dropped at a subsequent iteration.
Note as well that typically the penalty function's values $p(x^i)$ decrease when the iteration sequence approaches the feasible set D.
The results of numerical tests we are engaged in will be reported elsewhere.

Open Questions. 1. It is not yet known if there are interesting classes of multiextremal global optimization problems, where it can be shown that the number of constraints to be retained at each iteration is bounded.

2. An outer cut method with constraint dropping strategies that allows to use nonlinear cuts, conceptually could be applied for solving nonlinear programming problems having a large number of constraints: Solve the problem while considering only a small part of its constraints obtaining a solution x^1. Consider the constraints not yet taken into account that are violated by x^1 as potential cut functions and apply an outer cut procedure with constraint dropping strategy. Each subproblem actually consists in solving the original problem under a subset of its constraints.Can such a procedure be organized in a way to become competitive with other methods used in large-scale optimization?

4. Cutting Plane Methods.

Let in the outer cut concept D_1 be a polyhedral set and consider affine cuts l_k. Let

$$D = \{x \in \mathbb{R}^n : g(x) \leq 0\} \tag{14}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is convex (often: $g(x) = \max_{i \in I} g_i(x)$, where $g_i : \mathbb{R}^n \to \mathbb{R}$ is convex, $i \in I$, $I \subset \mathbb{N}$, $|I|$ finite).

Cutting plane methods for solving the resulting problems having convex objective function were given e.g. in [1],[6],[8],[9],[12],[17],[18], [23]. Concave programming was treated,e.g.,in [11], [13], [27], [29]. General global optimization concepts were treated in [14], [15].
Let the cuts l_k be defined by

$$l_k(x) = (p^k, x - y^k) + \beta_k, \tag{15}$$

where $p^k \in \mathbb{R}^n$, $y^k \in D_1$ and $\beta_k \in \mathbb{R}$ are suitably chosen. Note that well-known classical methods are obtained by specific choices of p^k, y^k, β_k. Set

$$p^k \in \partial g(y^k), \quad \beta_k = g(y^k), \tag{16}$$

where $\partial g(y^k)$ denotes the subdifferential (or ε-subdifferential, cf.[23]) of g at y^k.
Then, for $y^k = x^k$, we obtain the KCG procedure [1],[17]. The supporting hyperplane method results if y^k is chosen to be the intersection of the line segment $[y^0,x^k]$ joining an interior point y^0 and x^k with the boundary of D.

The general approaches of [14], [15] allow to construct a broad class of new cutting plane methods satisfying (15),(16) that include previous algorithms as special classes. Let conv f denote the convex hull of a set $A \subset \mathbb{R}^n$.

Theorem 4. Horst, Thoai, Tuy [15]) Let in problem (1) the feasible set D have the form (14) and assume that $D^0 := \{x \in \mathbb{R}^n : g(x) < 0\}$ is not empty. Let K be an arbitrary nonempty, convex, compact subset of D and choose in (15)

$$y^k \in \mathcal{Y}^k := \operatorname{conv}(K \cup \{x^k\}) \setminus D^0, \quad p^k \in \partial g(y^k) \setminus \{0\}, \quad \beta_k = g(y^k), \tag{17}$$

where $x^k \in \operatorname{argmin} f(D_k)$.
Then the separation requirements of the basic outer cut concept and the assumptions of Theorem 2 are satisfied.
The set $\mathcal{Y}^k$ is illustrated in Fig. 1.

Remark. Theorem 4 implies in particular that, whenever there is an algorithm for the relaxed problems (cf. Section 5), then the constraint dropping strategies resulting from Theorem 3 can be applied independently of the form of the objective function.

Open Question. It is not yet clear which one of the many implementable algorithms that fit with the result of Theorem 4 should be chosen for solving certain classes of global optimization problems. First preliminary computation on convex minimization suggests to choose $z^k \in D^0$ and to apply the supporting hyperplane method with y^0 replaced by z^k in such a way that the intersection point y^k of $[z^k, x^k]$ with the boundary of D approximately bisects $[z^k, y^k]$, i.e. $y^k \approx \frac{1}{2}(z^k + x^k)$.

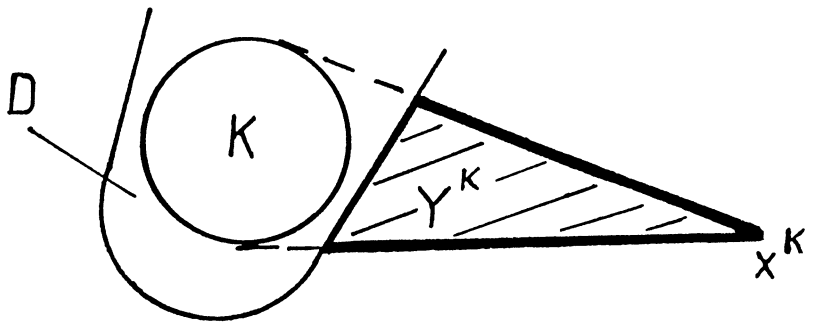


Fig. 1. The set Y^k of Theorem 4

5. On Solving the Relaxed Problems

5.1. New Vertices

In global optimization, outer cut methods have been applied for solving several types of problems, most of them belong to the class of concave progamming problems, where in (1) f is concave and D is convex and compact (cf.e.g.[7],[11],[13],[20],[21],[22],[25],[26],[29],[30],[31]). Except for very few examples (cf.e.g.[27]), the cuts always were linear i.e. cutting plane methods were applied.

One starts with a simple convex polytope D_1 that, in most cases, is an n-simples or an n-rectangle (for a discussion on how to find D_1, see e. g. [13]). Since f is assumed to be a concave function, we always have

$$\min f(D_k) = \min f(V(D_k)), \tag{18}$$

where $V(D_k)$ denotes the vertex set of D_k. Thus, the relaxed problems (P_k) are solved by comparing the objective function's values at the vertices of D_k. Then obviously, the computationally most expensive part of these algorithms consists in the calculation of the new vertices created by a cut:

Given the vertex set of a convex polytope P and given a hyperplane H created by the cutting plane algorithm, find the vertex set V(S) of $S := P \cap H$.

Since obviously, S is a convex polytope, one comes to the basic problem of finding all vertices of a convex polytope given by a system of finitely many inequalities. This problem has been discussed extensively in the literature and several algorithms are available (cf.e.g.[3],[4],[5], [19] and references given there.) However, numerous examples we calculated and the following considerations show that it is advisable to exploit specific given properties instead of applying one of the algorithms for finding all vertices of a convex polytope given by a system of linear inequalities. Let

$$H = \{ x \in \mathbb{R}^n : l(x) = (\alpha, x) + \beta = 0\}, \ \alpha \in \mathbb{R}^n, \ \beta \in \mathbb{R}. \tag{19}$$

Define

$$V^+(P) = \{v \in V(P) : h(v) > 0\}, \tag{20}$$

$$V^-(P) = \{v \in V(P) : h(v) < 0\}. \tag{21}$$

Note that, in contrast to the setting of the more general problem of finding all vertices of a polytope given by linear inequalities, in our case $V^+(P)$ and $V^-(P)$ are known. Moreover, for all examples calculated with a cutting plane procedure for solving concave minimization problems the number

$$\min \{|V^+(P)|, |V^-(P)|\}, \tag{22}$$

where $|V|$ denotes the number of elements of a finite set V, was always considerably smaller than the number $|V(S)|$ of generated vertices (for a detailed report on numerical experience, see [16]).

The algorithm we propose for finding V(S) is based on the following

Proposition 1. [16] We have that $w \in V(S)$ if and only if w is either a vertex of P lying in H or a point where an edge [u, v] of P, $u \in V^-(P)$, $v \in V^+(P)$ intersects H.

Let $|V| = \min \{|V^+(P)|, |V^-(P)|\}$. For each $u \in V$ denote by E(u) the set of halflines emanating from u, each of which contains an edge of P. The algorithm essentially consists in computing for each $e \in E(u)$, the intersection w of e with the hyperplane H and a feasibility check on $w \in P$ or $w \notin P$. Consider the constraints defining P and let J(u) be the index

set of those constraints that are active (binding) at u. For each system of n linearly independent constraints

$$(a^{i_k}, x) + b_{i_k} \leq 0, \; i_k \in J(u) \; (k = 1,\ldots,n) \tag{23}$$

add the new constraint $l(x) \leq 0$ or $l(x) \geq 0$ (depending on V). Then the necessary calculations can easily be carried out by pivoting on a simplex tableau corresponding to (23) and $l(x) \leq 0$ ($l(x) \geq 0$)) and a suitable bookkeeping. For details, see [16].
Note that, for nondegenerate vertices u, there is only one system (23). A detailed theoretical and numerical comparison with previous procedures proposed by Thieu, Tam and Ban [26] and by Falk and Hoffman [7],[11] is reported in [16]. It turned out that for very small problems with $|V(P)| \leq 100$, one should use our method or, in a very straightforward way, for each pair $u \in V^+(P)$, $v \in V^-(P)$, determine the intersection w of the line segment [u, v] with H and decide by the rank of the system of inequalities that are active at w, whether [u, v] is an edge of P or not. For larger problems our algorithm should be used.

5.2. Redundant Constraints.

Redundancy of constraints in linear programming has been discussed by many authors (cf. e.g. [5] and references given there). However,by considerations similar to those of the preceding subsection, in our case,a procedure for finding redundant constraints should be based on the knowledge of the vertex set V(P). A detailed presentation of suitable notions and characterizations of redundant constraints via V(P) is given in [16].
We briefly mention the most useful result.
Let

$$P = \{x \in \mathbb{R}^n : g_k(x) \leq 0, \; k \in I\} \tag{24}$$

be a convex polytope defined by the linear qualities $g_k(x) \leq 0$, $k \in I$, and let $\bar{P}$ be the convex polytope

$$\bar{P} = P \cap \{x \in R^n : l(x) \leq 0\}, \tag{25}$$

where $l(x) \leq 0$ is an additional linear constraint.

Definition 1. A constraint $g_k(x) \leq 0$, $k \in I$ is strictly redundant for the polytope P (relative to g_j) if there is a $j \in I \setminus \{k\}$ such that we have

$$F_k := P_j \cap \{x \in \mathbb{R}^n : g_k(x) = 0\} \subset \{x \in \mathbb{R}^n : g_j(x) > 0\}, \tag{26}$$

where $P_j := \{x \in R^n : g_i(x) \leq 0, \; i \in I \setminus \{j\}$.

Theorem 5. Horst,Thoai,de Vries [16]. a) A constraint $g_k(x) \leq 0$, $k \in I$, is strictly redundant for P relative to 1 if and only if

$$g_k(u) < 0 \quad \forall u \in V(P) \setminus V^+(P).$$

b) Every constraint that is strictly redundant for P (relative to any g_j) is strictly redundant for $\bar{P}$ relative to 1.

Remark. By Theorem 5, at an arbitrary chosen iteration of the cut procedure, all strictly redundant constraints can be identified. However, there can be redundant constraints that are not strictly redundant. A definition of redundancy is obtained, if in (26) the strict inequality $g_j(x) > 0$ is replaced by $g_j(x) \geq 0$. A characterization of redundancy relative to 1 that uses only known vertex sets is also given in [16]. However, if one wants to eliminate all constraints that are redundant but not strictly redundant by means of this characterization, one has to carry out the corresponding investigation at every iteration of the algorithm (cf. [16]). Therefore, in our cutting plane code for solving concave minimization, only strictly redundant constraints are eliminated in some iterations.

References.

[1] Cheney, E.W., Goldstein, A.A.: Newton's Method for Convex Programming and Tchebycheff Approximation. Numer. Math. 1, 253-268, 1959.

[2] Dantzig, G.B., Wolfe, P.: Decomposition Principle for Linear Programs, Oper. Res. 8, 101-111, 1960.

[3] Dyer, M.E.: The Complexity of Vertex Enumeration Methods. Math. Oper. Res. 8, 381-402, 1983.

[4] Dyer, M.E., Proll, L.G.: An Algorithm for Determining All Extreme Points of a Convex Polytope. Math. Programming 12, 81-96, 1977.

[5] Dyer, M.E., Proll, L.G.: An Improved Vertex Enumeration Algorithm. Europ. J. Oper. Res. 9, 359-368, 1982.

[6] Eaves, B.C., Zangwill, W.I.: Generalized Cutting Plane Algorithms. SIAM J. Control 9, 529-542, 1971.

[7] Falk, J., Hoffmann, K.R.: A Successive Underestimation Method for Concave Minimization Problems. Math. Oper. Res. 1, 251-259, 1976.

[8] Fukushima, M.: An Outer Approximation Algorithm for Solving General Convex Programs. Oper. Res. 31, 101-113, 1983.

[9] Fukushima, M.: On the Convergence of a Class of Outer Approximation mation Algorithms for Convex Programs. J. Comput. Appl. Math. 10, 147-156, 1984.

[10] Giannessi, F., Niccolucci, F.: Connections between nonlinear and integer programming problems. Symposia Mathematica XIX, Istituto Nazionale di Alta Math., Acad. Press N.Y., 161-178, 1976.

[11] Hoffman, K.R.: A Method for Globally Minimizing Concave Functions over Convex Sets. Math. Programming 20, 22-32, 1981.

[12] Hogan, W.W.: Applications of a General Convergence Theory for Outer Approximation Algorithms. Math. Programming 5, 151-168, 1973.

[13]Horst, R.: On the Global Minimization of Concave Functions: Introduction and Survey. Oper. Res. Spektrum 6, 195-205, 1984.

[14] Horst, R., Thoai, Ng.V., Tuy, H.: Outer Approximation by Polyhedral Convex Sets, forthcoming in Oper. Res. Spektrum.

[15] Horst, R., Thoai, Ng.V., Tuy, H.: On an Outer Approximation Concept. Arbeitsbericht Universität Trier, Fachbereich IV - Mathematik 8, 1987, submitted.

[16] Horst, R., Thoai, Ng. V., de Vries, J.: On Finding New Vertices and Redundant Constraints in Cutting Plane Algorithms for Global Optimization. Arbeitsbericht Universität Trier, Fachbereich IV - Mathematik 10, 1987, submitted.

[17] Kelley, J.E.: The Cutting-Plane Method for Solving Convex Programs. J. SIAM 8, 703-712, 1960.

[18] Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics 1133, Springer, 1985

[19] Matheis, T.H., Rubin, D.S.: A Survey and Comparison of Methods for Finding All Vertices of Convex Polyhedral Sets. Math. Oper. Res. 5, 167-184, 1980.

[20] Mayne, D.Q., Polak, E.: Outer Approximation Algorithm for Nondifferentiable Optimization Problems, J.Optim. Th. and Appl. 42, 19-30, 1984.

[21] Pardalos, P.M., Rosen, J.B.: Methods for Global Concave Minimization: A Bibliographic Survey. SIAM Review 28, 367-379, 1986.

[22] Pardalos, P.M., Rosen, J.B.: Constrained Global Optimization: Algo rithms and Applications, to appear in Lecture Notes in Computer Science, Springer, 1987.

[23] Parikh, S.C.: Approximate Cutting Planes in Nonlinear Programming. Math. Programming 11, 184-198, 1976.

[24] Raghavachari, M.: On Connections between Zero-One-Integer Programming and Concave Programming under Linear Constraints. Oper. Res. 17, 680-684, 1969.

[25] Thach, P.T., Tuy, H.: Global Optimization under Lipschitzian Constraints, forthcoming in Japanes J. Appl. Math. 1987.

[26] Thach, P.T., Tuy, Thoai, Ng.V.: Design Centering Problem with Lipschitzian Structure. Preprint, Institut of Mathematics, Hanoi, submitted.

[27] Thieu, T.V., Tam, B.T., Ban, V.T.: An Outer Approximation Method for Globally Minimizing a Concave Function over a Compact Convex Set. Acta Math. Vietnamica 8,1, 21-40, 1983.

[28] Topkis, D.M.: Cutting Plane Methods without Nested Constraint Sets. Oper. Res. 18, 404-413, 1970.

[29] Tuy, H.: On Outer Approximation Methods for Solving Concave Minimization Problems. Acta Math. Vietnamica 8,2, 3-34, 1983.

[30] Tuy, H.: Convex Programs with an Additional Reverse Convex Constraint. J. Optim. Th. and Appl. 52, 463-486, 1987.

[31] Tuy, H.: A General Deterministic Approach to Global Optimization via D.C.-Programming, in: Fermat Day 1985: Mathematics for Optimization, ed. J.B. Hiriart-Urruty, Mathematical Studies Series, North-Holland, 1986

[32] Veinott, A.F.: The Supporting Hyperplane Method for Unimodal Programming. Oper. Res. 15, 147-152, 1967.

GENERIC UNIQUENESS OF THE SOLUTION OF "MAX MIN" PROBLEMS

P. S. Kenderov
Institute of Mathematics
Bulgarian Academy of Sciences
"Acad. G. Bonchev"-Street, Block 8
1113 Sofia, Bulgaria

N. K. Ribarska

1. Introduction.

Let X and Y be compact spaces and f be from $C(X \times Y)$ (the space of all continuous real-valued functions on $X \times Y$). Consider the "max min" problem (P_f) of maximizing (over X) the function $\min \{ f(x,y) : y \in Y \}$. A solution to this problem is every pair $(x_0,y_0) \in X \times Y$ for which

$$f(x_0,y_0) = \min \{ f(x_0,y) : y \in Y \} = \max \{ \min \{ f(x,y) : y \in Y \} : x \in X \}$$

It is clear that every such problem has at least one solution. We show here that for a large class of compact spaces X and Y (including all metrizable and some nonmetrizable compacta) there exists a dense and G_δ subset A of $C(X \times Y)$ at each point of which the corresponding maxmin problem (P_f) has unique solution.

2. Definitions and notations.

First we remind some definitions.

2.1. Definition. A multivalued mapping $F : U_1 \longrightarrow U_2$ from the topological space U_1 into the topological space U_2 is said to be upper semi-continuous at the point $u_0 \in U_1$ if for every open set V in U_2 which contains $F(u_0)$ there exists a neighbourhood W of u_0

such that $F(u) \subset V$ for every $u \in W$. F is said to be upper semicontinuous if it is upper semi-contonuous at every point $u \in U_1$. The correspondence F is called *usco*, if it is upper semi-continuous and $F(u)$ is non-empty and compact for every $u \in U_1$.

2.2.Definition. A multivalued mapping $F : U_1 \longrightarrow U_2$ from a topological space U_1 into a topological space U_2 is said to have the property (*) at $u \in U_1$ if there exists a point w in $F(u)$ so that for each neighbourhood W of w there exists a neighbourhood V of u such that for every $v \in V$ the intersection $F(v) \cap W$ is non-empty.

In an implicit form this notion was used in [DK]. Explicitely it was involved in [Ch] and [ChK] where it was shown that for each *usco* mapping $F : U_1 \longrightarrow U_2$ from the complete metric space U_1 into the Eberlein or Radon-Nikodym (see [N]) compact space U_2 there exist a lot of points (forming a dense and G_δ subset) of U_1 at which F has the property (*). In [K3] this is also proved for *usco* mappings F with Gul'ko compact range space U_2.

2.3.Definition. A Hausdorff compact space X is said to belong to the class $\mathfrak{L}$ if for every complete metric space B and for every *usco* mapping $F : B \longrightarrow X$ there exists adense G_δ subset G of B such that for every $b \in G$ the mapping F has the property (*) at b.

The class $\mathfrak{L}$ has remarkable stability properties : it is closed under taking countable products, countable sums of closed sets and perfect images.

The idea to consider the class of all spaces X for which the property discussed in definition 2.3 is fulfilled belongs to Stegall [S1]. He considered the (eventually smaller) class of spaces X obtained if one replaces in definition 2.3 the words "complete metric" by "Baire". The properties of the class $\mathfrak{L}$ and of the spaces in it are discussed in [S1,S2,S3,S4,K1,K2,ČK1,ČK2]. We would like to mention here that $\mathfrak{L}$ contains (in increasing generality) all Eberlein compacta, all Talagrand compacta [D], all Gul'ko compacta [K3] and all fragmentable spaces [R].

3. "Max min" problems.

3.1. Theorem. Let X and Y be Hausdorff compacta and $X \times Y$ belongs to $\mathfrak{L}$. Then the set of all functions $f \in C(X \times Y)$, for which the corresponding "max min" problem (P_f) has unique solution, contains a dense G_δ subset of the space $C(X \times Y)$ equipped with the usual uniform convergence norm.

Proof.

Let us consider the multivalued mapping $S : C(X \times Y) \longrightarrow X \times Y$ assigning to each continuous function $f \in C(X \times Y)$ the set of solutions of (P_f), i.e.

$$S(f) = \left\{ (x_o,y_o) \in X \times Y : \begin{array}{l} f(x_o,y_o)=\max\{f(x_o,y): y\in Y\}= \\ \max\{ \min\{f(x,y): y\in Y\}: x\in X\} \end{array} \right\}$$

First we will show that $S : C(X \times Y) \longrightarrow X \times Y$ is an *usco* set-valued mapping. Next we show that S is single-valued at the points where it has the property (*). Since $X \times Y$ belongs to $\mathfrak{L}$, this will complete the proof.

3.2. Lemma. S is *usco*.

Proof of the lemma.

Since $X \times Y$ is compact, it suffices to prove that the graph of S is a closed subset of $C(X \times Y) \times (X \times Y)$.

We will show first that

(1) $\{ \min \{ f_\alpha(x_\alpha,y) : y\in Y \} \}_{\alpha\in A} \longrightarrow \min \{ f_o(x_o,y) : y\in Y \}$

provided $\{ f_\alpha \}_{\alpha\in A} \xrightarrow{\|.\|} f_o$ and $\{ x_\alpha \}_{\alpha\in A} \longrightarrow x_o$.

Indeed, let $\min \{ f_\alpha(x_\alpha,y) : y \in Y \} = f_\alpha(x_\alpha,y_\alpha)$ for some $y_\alpha \in Y$ and $\min \{ f_o(x_o,y): y \in Y \} = f_o(x_o,y_o)$ for some $y_o \in Y$. The compactness of Y yields a convergent subnet $\{ y_{\alpha_\beta} \}_{\beta\in B}$ of $\{ y_\alpha \}_{\alpha\in A}$. Let $\tilde{y}$ be its limit. Then

(2) $f_{\alpha_\beta}(x_{\alpha_\beta},y_o) \geq \min \{ f_{\alpha_\beta}(x_{\alpha_\beta},y) : y \in Y \} = f_{\alpha_\beta}(x_{\alpha_\beta},y_{\alpha_\beta})$.

But

$$| f_{\alpha_\beta}(x_{\alpha_\beta}, y_{\alpha_\beta}) - f_o(x_o, \tilde{y}) | \leq$$

$$|f_{\alpha_\beta}(x_{\alpha_\beta}, y_{\alpha_\beta}) - f_o(x_{\alpha_\beta}, y_{\alpha_\beta})| + |f_o(x_{\alpha_\beta}, y_{\alpha_\beta}) - f_o(x_o, \tilde{y})| \leq$$

$$\| f_{\alpha_\beta} - f_o \| + | f_o(x_{\alpha_\beta}, y_{\alpha_\beta}) - f_o(x_o, \tilde{y}) |$$

and hence $\{ f_{\alpha_\beta}(x_{\alpha_\beta}, y_{\alpha_\beta}) \}_{\beta \in B} \longrightarrow f_o(x_o, \tilde{y})$. Analogously $\{ f_{\alpha_\beta}(x_{\alpha_\beta}, y_o) \}_{\beta \in B} \longrightarrow f_o(x_o, y_o)$. From the inequality (2) we obtain

$$f_o(x_o, y_o) \geq f_o(x_o, \tilde{y}).$$

On the other hand

$$f_o(x_o, \tilde{y}) \geq \min \{ f_o(x_o, y) : y \in Y \} = f_o(x_o, y_o).$$

We conclude that

$$f_o(x_o, \tilde{y}) = \min \{ f_o(x_o, y) : y \in Y \}$$

and so

$$\{ \min \{ f_{\alpha_\beta}(x_{\alpha_\beta}, y): y \in Y \} \}_{\beta \in B} \longrightarrow \min \{ f_o(x_o, y): y \in Y \}$$

We obtained that every subnet of $\min \{ f_\alpha(x_\alpha, y): y \in Y \}$ has a subsubnet converging to $\min \{ f_o(x_o, y) : y \in Y \}$. Hence (1) holds.

Let $\{ (f_\alpha, x_\alpha, y_\alpha) \}_{\alpha \in A}$ is a covergent net contained in the graph of S $Gr(S)$ (i.e. $(x_\alpha, y_\alpha) \in S(f_\alpha)$ for every $\alpha \in A$). Let us denote its limit by (f_o, x_o, y_o). Then

$$f_\alpha(x_\alpha, y_\alpha) = \min \{ f_\alpha(x_\alpha, y): y \in Y \} \geq \min \{ f_\alpha(x, y): y \in Y \}$$

for every $\alpha \in A$ and every $x \in X$. Using (1) we conclude that

$$f_o(x_o, y_o) = \min \{ f_o(x_o, y): y \in Y \} \geq \min \{ f_o(x, y): y \in Y \}$$

for every $x \in X$, i.e. $(x_o, y_o) \in S(f_o)$ or $(f_o, x_o, y_o) \in Gr(S)$. Therefore $Gr(S)$ is a closed subset of $C(X \times Y) \times (X \times Y)$ and, because of the compactness of the range space $X \times Y$, S is upper semi-continuous. It is evident that S has non-empty compact images because of (1).■

3.3.Lemma. Let S have the property (*) at $f_o \in C(X \times Y)$ and (x_o, y_o) be the point mentioned in (*), i.e. for every neighbourhood U of (x_o, y_o) there exists $\varepsilon > 0$ so that the intersection $S(f) \cap U$ is nonempty whenever $f \in C(X \times Y)$ and $\| f - f_o \| < \varepsilon$. Then the set $S(f_o)$ has only one point, i.e. $S(f_o) = \{ (x_o, y_o) \}$.

Proof of the lemma.

Let us suppose that $S(f_0)$ contains more than one element. We will consider two cases.

Case 1. There exists a point $(x_1,y_1) \in S(f_0)$ such that $x_0 \neq x_1$. Then there exits a continuous function a defined on X such that $\| a \|_{C(X)} = 1$, $a(x_1) \leq 0$ and $a(x_0) > 0$. Let us denote by V the open set $\{ x \in X : a(x) > 0 \} \times Y \subset X \times Y$. For every $\varepsilon > 0$ we consider the function $f_\varepsilon \in C(X \times Y)$ defined by $f_\varepsilon(x,y) = f_0(x,y) - \varepsilon.a(x)$. It is clear that $\| f_0 - f_\varepsilon \| < \varepsilon$. We will show that $S(f_\varepsilon) \cap V = \emptyset$ which will contradict the property (*).

Let $(\tilde{x},\tilde{y})$ be an arbitrary point of V. We have

$\min \{ f_\varepsilon(\tilde{x},y) : y \in Y \} =$

$\min \{ f_0(\tilde{x},y) : y \in Y \} - a(\tilde{x}) \leq v_0 - \varepsilon.a(\tilde{x}) < v_0$,

where $v_0 = \max \{ \min \{ f_0(x,y) : y \in Y \} : x \in X \}$.

On the other hand, since $(x_1,y_1) \in S(f_0)$, we have

$v_0 = \min \{ f_0(x_1,y) : y \in Y \} \leq$

$\min \{ f_\varepsilon(x_1,y) : y \in Y \} \leq$

$\max \{ \min \{ f_\varepsilon(x,y) : y \in Y \} : x \in X \}$.

Therefore

$\min \{ f_\varepsilon(\tilde{x},y) : y \in Y \} < v_0 \leq$

$\max \{ \min \{ f_\varepsilon(x,y) : y \in Y \} : x \in X \}$

and hence $(\tilde{x},\tilde{y}) \notin S(f_\varepsilon)$.

Case 2. There exists a point $y_1 \in Y$, $y_1 \neq y_0$, for which $(x_0,y_1) \in S(f_0)$. Hence there exists a continuous function b on Y with $\| b \|_{C(Y)} = 1$, $b(y_1) \leq 0$ and $b(y_0) > 0$. We denote by V the open set $X \times \{ y \in Y : b(y) > 0 \} \subset X \times Y$.

Let $\varepsilon > 0$ be an arbitrary positive number. Consider the function $g_\varepsilon : X \times Y \longrightarrow \mathbb{R}$ defined by

$$g_\varepsilon(x,y) = \begin{cases} f_0(x,y) + \varepsilon/5 & \text{if } f_0(x,y) < v_0 - \varepsilon/5 \\ v_0 & \text{if } f_0(x,y) \in [\, v_0 - \varepsilon/5,\ v_0 + \varepsilon/5 \,] \\ f_0(x,y) - \varepsilon/5 & \text{if } f_0(x,y) > v_0 + \varepsilon/5 \end{cases}$$

where $v_0 = \max \{ \min \{ f_0(x,y) : y \in Y \} : x \in X \}$.

It is streightforward to check that g_ε is continuous on $X \times Y$, $\| g_\varepsilon - f_o \| \leq \varepsilon/5$ and

$$v_o = \min \{ g_\varepsilon(x_o,y) : y \in Y \} = \max \{ \min \{ g_\varepsilon(x,y) : y \in Y \} : x \in X \}.$$

The compactness of Y and $f_o(x_o,y) > v_o - \varepsilon/5$ for every $y \in Y$ yield the existence of a neighbourhood U'_ε of x_o for which $f_o(x,y) > v_o - \varepsilon/5$ for every $x \in U'_\varepsilon$ and for every $y \in Y$. Let U''_ε be a neighbourhood of x_o for which $f_o(x,y_1) < v_o + \varepsilon/5$ whenever $x \in U''_\varepsilon$. We set $U_\varepsilon = U'_\varepsilon \cap U''_\varepsilon$. Thus U_ε is a neighbourhood of x_o with $g_\varepsilon(x,y_1) = v_o \leq g_\varepsilon(x,y)$ for every $x \in U_\varepsilon$ and for every $y \in Y$. Consider the function $f_\varepsilon: X \times Y \longrightarrow \mathbb{R}$ defined by

$$f_\varepsilon(x,y) = g_\varepsilon(x,y) + \varepsilon/5.b(y) + 3\varepsilon/5.a_\varepsilon(x)$$

where a_ε is a continuous function on X with $a_\varepsilon(x_o) = 1$, $a_\varepsilon(x) = 0$ for every $x \notin U_\varepsilon$ and $\| a_\varepsilon \|_{C(X)} = 1$.

It is evident that f_ε is continuous and

$$\| f_o - f_\varepsilon \| \leq \varepsilon/5 + \varepsilon/5 + 3\varepsilon/5 = \varepsilon.$$

We will show that $S(f_\varepsilon) \cap V = \emptyset$ for every $\varepsilon > 0$. This will contradict the property (*).

Let $(\tilde{x},\tilde{y})$ be a point in V. If $\tilde{x}$ belongs to U_ε we have $g_\varepsilon(\tilde{x},y_1) \leq g_\varepsilon(\tilde{x},\tilde{y})$ by the construction of U_ε and hence $f_\varepsilon(\tilde{x},y_1) < f_\varepsilon(\tilde{x},\tilde{y})$, because $b(y_1) < b(\tilde{y})$ for $(\tilde{x},\tilde{y})$ in V. Then

$$f_\varepsilon(\tilde{x},\tilde{y}) \neq \min \{ f_\varepsilon(\tilde{x},y) : y \in Y \}$$

and so $(\tilde{x},\tilde{y})$ doesn't belong to $S(f_\varepsilon)$.

If $\tilde{x} \notin U_\varepsilon$, then $a_\varepsilon(\tilde{x}) = 0$ and we have

$$\min \{ f_\varepsilon(\tilde{x},y) : y \in Y \} =$$

$$\min \{ g_\varepsilon(\tilde{x},y) + \varepsilon/5.b(y) : y \in Y \} \leq$$

$$\min \{ g_\varepsilon(\tilde{x},y) : y \in Y \} + \varepsilon/5 \leq$$

$$v_o + \varepsilon/5,$$

On the other hand

$$\min \{ f_\varepsilon(x_o,y) : y \in Y \} =$$

$\min \{ g_\varepsilon(x_o,y) + \varepsilon/5.b(y) : y \in Y \} + 3\varepsilon/5 \geq$

$\min \{ g_\varepsilon(x_o,y) : y \in Y \} - \varepsilon/5 + 3\varepsilon/5 =$

$v_o + 2\varepsilon/5.$

Therefore

$\min \{ f_\varepsilon(\tilde{x},y) : y \in Y \} \leq v_o + \varepsilon/5 <$

$v_o + 2\varepsilon/5 \leq \min \{ f_\varepsilon(x_o,y) : y \in Y \}$

and hence $(\tilde{x},\tilde{y})$ doesn't belong to $S(f_\varepsilon)$.■

Theorem 3.1 is proved.

REFERENCES

[Ch] H.P.R.Christensen, Theorems of Namioka and R.E.Johnson type for upper semicontinuous and compact valued set-valued mappings, PAMS, Vol.86, No.4(1982),649-655.

[ChK] H.P.R.Christensen, P.S.Kenderov, Dense strong continuity of mappings and the Radon-Nikodym property, Univ. Copenhagen Preprint Ser.#17, 1982; published in Math. Scand. 54(1984), 70-78.

[ČK1] M.M.Čoban, P.S.Kenderov, Generic Gateaux differentiability of convex functionals in C(T) and the topological properties of T, C.R.Acad. Bulgare Sci. 38(1985), 1603-1604.

[ČK2] M.M.Čoban, P.S.Kenderov, Generic Gateaux differentiability of convex functionals in C(T) and the topological properties of T, Math. and Education in Math. 1986, Proc. 15-th Conf. Union of Bulg. Mathematicians, Sunny beach, April 1986,141-149.

[D] G.Debs, Espaces K-analytiques et espaces de Baire de fonctions continues, Mathematika 32(1985), 218-228.

[DK] F.Deutsch, P.S.Kenderov, Continuous selections and approximate selections for set-valued mappings and applications to metric projections, SIAM J. Math. Anal., Vol.14, No.1, 1983.

[K1] P.S.Kenderov, Most of the optimization problems have unique solution, C.R.Acad. Bulgare Sci. 37,#3(1984).

[K2] P.S.Kenderov, Most of the optimization problems have unique solution, International Series of Numerical Mathematics, Vol.72 (1984), Birkhäuser Verlag Basel, 203-216.

[K3] P.S.Kenderov, C(T) is weak Asplund for every Gul'ko compact T, C.R.Acad. Bulgare Sci. 40,#2(1987).

[N] I.Namioka, Radon-Nikodym Compact Spaces and Fragmentability, preprint.

[R] N.K.Ribarska, Internal characterization of fragmented spaces, to appear.

[S1] C. Stegall, A class of topological spaces and differentiation of functions on Banach spaces, Vorlesungen aus dem Fachbereich Mathematik der Universitat Essen, 1983.

[S2] C. Stegall, More Gateaux differentiability spaces, in Banach spaces, Proceedings, Missouri 1984, Springer Lecture Notes no.1166, Berlin, 1985.

[S3] C. Stegall, Topological spaces with dense subspaces that are homeomrphic to complete metric spaces and the classification of C(K) Banach spaces, preprint.

Optimal stopping and leavable gambling models with observation costs.

Manfred Schäl, Universität Bonn, Institut für Angewandte Mathematik,
Wegelerstr. 6, D-5300 Bonn 1.

Summary. It is well-known that for the problem of stopping a Markov chain with finite state space there exists an optimal a.s. finite stopping time which is the entrance time into the set where the value function coincides with the utility function. In this paper, this result is extended to the case where more than one continue action is available. The result of the paper also yields a sufficient condition for the existence of a stationary optimal policy in an extended leavable gambling house with compact action space where observation costs are incurred.

Key words and phrases: Dynamic programming, gambling, optimal stopping, average return, limit supremum payoff, stationary policy, finite stopping times.

§ 1 Introduction.

We consider a system with a finite number of states $i \in S$. Periodically we observe the current state of the system and then choose an action a from a set A of possible actions. As a result of the current state i and the chosen action a, the system moves to a new state j with probability $p_{ij}(a)$. As a further consequence, an immediate cost $c(i,a)$ is incurred. Our desires are summerized, in a certain sense, by a function u from S to real numbers, called the utility function. The worth of the state i is $u(i)$. If $h = (i_o, i_1, i_2, \ldots)$ is a possible history of the system which stagnates at time n, i.e., $i_n = i_{n+1} = i_{n+2} = \ldots$, then the history describes the case where no observations are made after time n. Such a history is valued according to the utility $u(i_n)$ of the terminal state minus the observation costs $\{c(i_o,a_o) + \ldots + c(i_{n-1},a_{n-1})\}$ where a_t is the action chosen at time t.

If the history does not stagnate at some finite epoch n, then it is no longer obvious how to value the history. In the framework of gambling models (cp. Dubins & Savage (1965), Sudderth (1971)) and optimal stopping models (Chow, Robbins & Siegmund (1971)), the history is valued according to

$$(1.1) \quad \overline{\lim}_{n\to\infty} \{u(i_n) - \Sigma_{t=o}^{n-1} c(i_t,a_t)\},$$

where however $c = o$ in the gambling literature. Actually, for an infinite state space which is not considered here, this definition is essential in order to get optimal possibly infinite stopping times for stopping problems. Then a policy which specifies how to choose the actions is valued according to the expectation of (1.1).

In dynamic programming with the total cost criterion, the optimality criterion for policies is given in a natural way. There $u=o$ and $\Sigma_{t=o}^{\infty} c(i_t,a_t)$ exists; moreover the sum and the expectation can be permuted by virtue of the usual convergence assumptions. However, in the case of the present paper, these assumptions are not satisfied mainly because $u(i_n)$ will generally not converges for non-stagnating histories. Therefore we will borrow methods from dynamic programming with the average cost criterion which also deals with non-convergent sequences. There one make use of the more general concept of the limit in the sense of Cesaro and Abel.

The present paper has predecessors. In Schäl (1985), the case $c=o$ is treated. There, results for the average cost criterion directly apply. In Schäl (1986) the present situation with observation cost is treated

for a Cesaro criterion and an Abel criterion. The purpose of the present paper is to strenghthen the latter results by extending them to the limit supremum criterion (1.1). There we can built on results by Schäl (1986) and on methods for gambling models with observation cost (or equivalently: with intermediate one-step rewards) presented in Schäl (1986a).

§ 2 The model

The underlying model is given by a tupel (S,A,p,c,u).

(i) S stands for the state space and is assumed to be finite.

(ii) A is the action space.

(iii) $p_{ij}(a)$ are the transition probabilities, where $p_{ij}(a) \geq 0$ and $\sum_{j\in S} p_{ij}(a) = 1$ for all $i\in S, a\in A$.

(iv) $c(i,a)$ is the real valued cost function.

(v) $u(i)$ is the real valued utility function.

The results will easily extend to the case where the set A of available actions depends on the current state.

We write $H_t = S\times\ldots\times S$ ($t+1$ factors) for the set of all histories $(i_o,\ldots,i_t)$ at stage t. A policy $\delta = (\delta_t)$ is defined as a sequence of decision functions $\delta_t : H_t \to A$. We write Δ for the set of all policies. An important role plays the set $\mathbb{F}$ of stationary policies. They can be identified with decision functions $f : S \to A$ where $\delta \hat{=} f$ iff $\delta_t(\ldots,i_t) = f(i_t)$ for all t. Given the initial state i, any policy $\delta\in\Delta$ defines a stochastic process (X_n) where X_n describes the state of the system at stage n. The process (X_n) is a Markov chain if the system is controlled by a stationary policy $f\in\mathbb{F}$. Let us write $X_{(n)} = (X_o,\ldots,X_n)$ for the history up to stage n and

$$(2.1)\quad u_n(\delta) = u(X_n) - \sum_{t=o}^{n-1} c(X_t,\delta_t(X_{(t)}))$$

for the gain if we stop at stage n.

The cost function may be negative and is then to be interpreted as reward. However, we assume that the total expected cost is bounded from below in the following sense:

Assumption (B): $\quad \sup_{\delta\in\Delta} E_{i\delta}\,[\sum_{t=o}^{\infty} c^-(X_t,\delta_t(X_{(t)}))] < \infty \quad$ for all $i \in S$.

This is the weakest assumption which is made in dynamic programming with the total cost criterion where $u=o$, and (B) defines then the so-called quasi-convergent case. However, our model is not a special case of this quasi-convergent case because of the utility function u and needs more general methods.

For any stopping time τ with respect to the process (X_n) which is

$P_{i\delta}$ - a.s. finite, i.e., $P_{i\delta}(\tau<\infty)<\infty$, in particular for a constant $\tau=n$ the expected gain up to stage τ is now well-defined as

(2.2) $v_{\tau\delta}(i) = E_{i\delta}[u_\tau(\delta)]$,

since u is bounded and

(2.3) $u_n(\delta) \leq \sup u + \Sigma_{t=0}^{\infty} c^-(X_t, \delta_t(X_{(t)}))$.

If τ is not finite, e.g. if $\tau = \infty$ everywhere, the definition of the expected gain is more problematic. For this paper we will use the following definition which essentially stems from the theory of gambling:

(2.4) $v_\delta(i) = E_{i\delta}[\overline{\lim}_{n\to\infty} u_n(\delta)]$.

Again, by (B), v_δ is well-defined. Now, the value function is

(2.5) $v(i) = \sup_{\delta\in\Delta} v_\delta(i)$.

Remark. Assumption (B) may even be replaced by the following weaker assumptions: v_δ is well-defined, $v<\infty$, $\{(u_n(\delta))^+\}$ is uniformly $P_{i\delta}$-integrable for all $i\in S, \delta\in\Delta$, $\{(u_n(\delta))^-\}$ is uniformly $P_{i\delta}$-integrable for all $i\in S, \delta\in\Delta$, such that $v_\delta(i) > -\infty$. □

We want to incorporate stopping into the policy. Therefore we assume that the underlying model is leavable, i.e. we make the following general assumption.

Assumption (A): $A = A_o + \{e\}$ where $p_{ij}(e) = \delta_{ij}$ and

$c(i,e) = 0$ for all $i,j \in S$.

Choosing action e means that one pause for one period, i.e., the system remains in the same state and no cost is incurred. As Dubins & Savage (1965), we interpret stopping at time τ by continuation such that one always chooses action e from stage τ onwards. Then, with probability one, the system remains in the state of stage τ time after time and $u_\tau(\delta) = \overline{\lim}\, u_n(\delta)$.

2.6 Lemma. Let $\delta \in \Delta$ and τ be a stopping time such that

$P_{i\delta}(\tau<\infty) = 1$ and $\delta_t(X_{(t)}) = e$ for $t \geq \tau$, then $v_\delta(i) = v_{\tau\delta}(i)$.

In the case where A_o is a singleton set, one has an ordinary stopping model.

2.7 Lemma. $u \leq v < \infty$.

Proof. For the stationary policy f where $f(i) = e$ for all $i\in S$ one

has $v_f = u$, hence $u \leq v$. From (B) and (2.3) we get $v < \infty$. □
In addition to (2.3) we need an integrable minorant for $\{u_n(\delta)\}$. Now

(2.8) $u_n(\delta) \geq \inf u - \sum_{t=0}^{\infty} c^+(X_t, \delta_t(X_{(t)}))$.

Integrability follows from

2.9 Lemma. $E_{i\delta}[\sum_{t=0}^{\infty} c^+(X_t, \delta_t(X_{(t)}))] < \infty$ if $v_\delta(i) > -\infty$.

Proof. Let us write $c_t = C(X_t, \delta_t(X_{(t)}))$. Then

$\sum_{t=0}^{n-1} c_t^+ = \sum_{t=0}^{n-1} c_t^- + u(X_n) - u_n(\delta)$. Hence,

$\sum_{t=0}^{\infty} c_t^+ \leq \sum_{t=0}^{\infty} c_t^- + \sup u - \overline{\lim}\, u_n(\delta)$

by taking $\underline{\lim}$ on both sides. Now (B) applies. □

§ 3 Relation to the Cesaro-Abel-criterion.

Another definition of the gain can be based on the limits in the sense of Cesaro and Abel.

(3.1) $v_\delta^C = \overline{\lim}_{n\to\infty} \frac{1}{n} (v_{1\delta} + \ldots + v_{n\delta})$

(3.2) $v_\delta^A = \overline{\lim}_{\beta\uparrow 1} (1-\beta) \sum_{n=0}^{\infty} \beta^n v_{n\delta}$, $\delta \in \Delta$.

3.3 Lemma. $v_\delta^A \leq v_\delta^C \leq v_\delta$ for $\delta \in \Delta$.

Proof. By (B) and (2.3), the sequences $\{v_{n\delta}(i)\}$ is bounded from above. Now the first inequality follows from a well-known relation between the Cesaro and the Abel limits (cp. Hordijk (71)). Again by (B) and (2.3), Fatou's Lemma applies and one obtains

(3.4) $\overline{\lim}\, v_{n\delta}(i) \leq v_\delta(i)$, $i \in S$.

Now the second inequality is obvious. □

From the theory of Markov chains one knows that the limits in (3.1), (3.2) exist and coincide for stationary policies $f \in \mathbb{F}$. Therefore we write

(3.5) $w_f = v_f^C = v_f^A$, $f \in \mathbb{F}$.

(3.6) $w = \sup_{f \in \mathbb{F}} w_f$.

From the argument for 2.7 and from 3.3 we obtain

(3.7) $u \leq w \leq v < \infty$.

Now let $\mathbb{P}_f$ be the transition operator for the Markov chain under f; i.e. for any real-valued function r on S one has

(3.8) $\mathbb{P}_f r(i) = \sum_{j \in S} p_{ij}(f(i)) r(j)$, $i \in S$,

and let c_f be defined by $c_f(i) = c(i,f(i))$, Then clearly

(3.9) $v_{of} = u$, $v_{n+1,f} = \mathbb{P}_f v_n - c_f$

and as a consequence since the limits in (3.5) exist:

(3.10) $w_f = \mathbb{P}_f w_f - c_f$.

3.10 Proposition. (a) For $\varepsilon > 0$ there exists some $f \in \mathbb{F}$ such that

$$w_f \geq w - \varepsilon .$$

(b) (superharmonicity)

$$w(i) \geq \sum_{j \in S} p_{ij}(a) w(j) - c(i,a) , \quad i \in S , a \in A .$$

Proof. a) By definition, one has for $\varepsilon > 0$ and $i \in S$ some $f_i \in \mathbb{F}$ such that $w_{f_i}(i) \geq w(i) - \varepsilon$. Now consider a model where A_o is replaced by $\{f_i(j) ; i,j \in S\} = \tilde{A}_o$. The policies f_i are admissible for this submodel. By the finiteness of $\tilde{A}_o$ we know from the results of Schäl (1986) that there exists an optimal stationary f for the submodel. Hence $w_f(i) \geq w_{f_i}(i) \geq w(i) - \varepsilon$.

b) Let be $\varepsilon > 0$, f as in part (a), and $\varphi \in \mathbb{F}$ arbitrary. Now consider a submodel where A_o is replaced by $\{f(i), \varphi(i), i \in S\}$. For this submodel the non-stationary policy $\delta = (\varphi,f,f,\ldots)$ is admissible. For δ , similarly to (3.5) and (3.10), the limits in (3.1),(3.2) exist and coincide and one has $v^C_\delta = v^A_\delta = \mathbb{P}_\varphi w_f - c_\varphi$. Let f_o be optimal for the submodel. Then

$$w \geq w_{f_o} \geq v^A_\delta = \mathbb{P}_\varphi w_f - c_\varphi \geq \mathbb{P}_\varphi w - c_\varphi - \varepsilon . \quad \square$$

Now define similarly to (2.1):

(3.11) $w_n(\delta) = w(X_n) - \sum_{t=o}^{n-1} c(X_t, \delta_t(X_{(t)}))$.

3.12 Lemma. Let be $\delta \in \Delta, i \in S$ such that $v_\delta(i) > -\infty$. Then $\{w_n(\delta)\}$ is a supermartingale, i.e.,

$$E_\delta[w_{n+1}(\delta) | X_{(n)}] \leq w_n(\delta) \quad \text{for all } n .$$

Moreover, $\lim w_n(\delta)$ exists and is $P_{i\delta}$-integrable where

$$E_{i\delta} \lim w_n(\delta) \leq w(i).$$

Proof. A relation similar to (2.3) shows that $(w_n(\delta))^+$ is integrable. Since $w \geq u$ and $v_\delta(i) > -\infty$, $w_n(\delta) \geq u_n(\delta)$ and $\{w_n(\delta)\}$ has a $P_{i\delta}$-integrable minorant by (2.8) and 2.9. Now for $h = (\ldots,i) \in H_n$

$$E_\delta[w(X_{n+1}) - c(X_n, \delta_n(X_{(n)})) | X_{(n)} = h]$$

$= \sum_{j \in S} p_{ij}(\delta_n(h))w(j) - c(i,\delta_n(h))$

$\geq w(i)$ by 3.10b

which shows the supermartingale property.

By the martingale convergence theorem, $\lim w_n(\delta)$ exists a.s. and is $P_{i\delta}$-integrable. By Fatou's Lemma

$E_{i\delta} \lim w_n(\delta) \leq \lim E_{i\delta}\, w_n(\delta) \leq E_{i\delta}\, w_o(\delta)$

since $\{E_{i\delta}\, w_n(\delta)\}$ is decreasing. Finally $w_o(\delta) = w(X_o)$. □

3.13 Theorem. $w = v$.

Proof. Let be $i \in S$, $\delta \in \Delta$ such that $v_\delta(i) > -\infty$. Then by 3.12

$w(i) \geq E_{i\delta} \lim w_n(\delta) \geq E_{i\delta} \overline{\lim}\, u_n(\delta) = v_\delta(i)$

since $w \geq u$. Hence $w \geq v$. But $w \leq v$ is known from (3.7). □

3.14 Corollary. Let be $f \in \mathbb{F}$. If $w_f = w$, then $w_f = v_f = v$.

Proof. From 3.3 and 3.13 one knows that

$w = w_f \leq v_f \leq v = w$. □

§ 4 Optimal policies.

In order to obtain optimal policies we assume compactness and continuity assumptions.

Assumption (C): (1) A_o is a compact metric space and e is an isolated point.

(2) $a \to p_{ij}(a)$ is continuous for all i,j.

(3) $a \to c(i,a)$ is lower semi-continuous for all i.

As in Schäl (85), (86) we need an additional continuity assumption. We premise the following lemma which is easy to prove under (C).

3.1 Lemma. The following conditions are equivalent for all i,j.:

(i) $\{a \in A_o\ ;\ p_{ij}(a) > o\}$ is closed ;

(ii) $a \to \operatorname{sign} p_{ij}(a)$ is is continuous on A_o ;

(iii) $\inf \{p_{ij}(a)\ ;\ a \in A_o\ ,\ p_{ij}(a) > o\} > o$.

Assumption (C*): For all $i,j \in S$ one of the conditions in Lemma 2.1 is satisfied.

Two important examples where (C*) is satisfied are the cases where A_o is finite or where (for all $i,j \in S$) $p_{ij}(a) = 0$ either for all $a \in A_o$ or for none.

On combining Corollary 3.14 with results from Schäl (1986, Theorem 2.4, Corollary 2.5) one obtains:

3.2 Theorem. There exists an optimal stationary policy f, i.e., $v_f = v$, such that

$$f(i) = e \quad \text{if} \quad v(i) = u(i)$$

$$f(i) \in A_o \quad \text{if} \quad v(i) \neq u(i).$$

Moreover $v_f = v_{\tau f}$ where τ is the entrance time into the set $[v = u]$ and $E_{if}\,\tau < \infty$ for all $i \in S$.

In Schäl (1986) continuity of $a \to c(i,a)$ was assumed rather than semi-continuity. However the proof given there can be extended to the case of the present paper (cp. Schäl (1986a, §17)).

References.

1. Chow, Y.S., Robbins,H., Siegmund,D. (1971): Great Expectations: The theory of optimal stopping.
   Boston: Houghton Mifflin Company.
2. Dubins,L.E., Savage, L.J. (1965): How to gamble if you must.
   New York: McGraw-Hill.
3. Hordijk,A. (1971): A sufficient condition for the existence of an optimal policy with respect to the average cost criterion in Markovian decision processes.
   Transactions of the 6th Prague Conference on Information Theory, Statistical Decision Functions, Random Processes, 263 - 274.
4. Schäl, M. (1985): Optimal stopping and leavable gambling models with the average return criterion.
   Contributions to Operations Research, K. Neumann & D. Pallaschke (eds). Lecture Notes in Economics and Math.Systems 240, 174 - 181.
5. Schäl, M. (1986): Markov and semi-Markov decision models and optimal stopping.
   Semi-Markov Models, ed. J.Janssen, Plenum Press, New York-London 39 - 62
6. Schäl, M. (1986a): Markoffsche Entscheidungsprozesse. Vorlesungsreihe SFB 72, no.33, Inst.Angew.Math., Univ. Bonn.
7. Sudderth, W.D. (1971): On measurable gambling problems.
   Ann.Math.Statist. 42, 260 - 269.

MULTI-RATIO FRACTIONAL PROGRAMMING - A SURVEY

Siegfried Schaible
Faculty of Business
University of Alberta
Edmonton, Alberta
T6G 2R6 Canada

Abstract

Nonlinear programming problems are considered where the objective function involves several ratios. We review recent results concerning three classes of multi-ratio fractional programs: 1) maximization of a sum of ratios, 2) maximization of the smallest of several ratios, and 3) multiobjective fractional programs. In addition to these results open problems are addressed as well.

Introduction

The large majority of articles in fractional programming deals with the optimization of a single ratio. Until 1980 only a few isolated results had appeared on multi-ratio fractional programs. Interestingly enough one of the earliest publications contributing to fractional programming, J. von Neumann's classical paper on a model of a general economic equilibrium in 1937 [26, 27], contains an analysis of a multi-ratio fractional program. In this early paper even a duality theory for this nonconvex program had been suggested, at a time when linear programming was hardly known.

Even at the recent NATO Advanced Study Institute on "Generalized Concavity in Optimization and Economics" [24] in 1980 most papers in fractional programming contributed to single-ratio fractional programming. However, since that time a good number of results have appeared in multi-ratio fractional programming. A new area of research has been opened up.

Most of the contributions deal with one of the following types of multi-ratio fractional programs: 1) maximization of a sum of ratios, 2) maximization of the smallest of several ratios, and 3) multiobjective fractional programs. In this survey we report on results for each of these classes of fractional programs. We mainly focus on results in the derivation of which the author has been personally involved.

1. Maximizing a Sum of Ratios

Consider

$$(1.1) \qquad \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \to \text{Sup}$$
$$\text{subject to}$$
$$x \in S$$

where $S \subseteq R^n$ is a convex feasible region. In single-ratio fractional programming

the classical assumption often met in applications is that the numerator is nonnegative and concave and the denominator is positive and convex. Then a local maximum is always a global maximum, duality relations can be established and several algorithms are available to solve such concave-convex single-ratio fractional programs [19]. Unfortunately, for multi-ratio fractional programs (1.1) none of these properties is true any longer if the classical assumption on f_i/g_i is satisfied [18]. In particular, a local maximum may not be a global one, even if a simple function like the sum of a linear ratio and a linear function is considered. Almost no results are known yet about the properties of (1.1) and how to solve this problem efficiently if f_i/g_i are concave-convex.

On the other land, some limited progress could be made for the maximization of the sum of certain non-concave-convex ratios and a concave or convex function [21]. Such a problem arises when the sum of a relative and absolute term is to be maximized. It turns out that the objective function is often quasiconcave or quasiconvex [2] if f_i/g_i is <u>not</u> concave-convex but satisfies other kinds of convexity assumptions [21].

Another example of model (1.1) which is tractable and where the classical assumptions on f_i/g_i are <u>not</u> satisfied is discussed in [23]. The model analyzed in this paper is designed to determine simultaneously optimal lot sizes and an optimal storage allocation in a warehouse. The total cost to be minimized is

$$(1.2) \qquad K(x) = \sum_{j=1}^{n} \left(\frac{\alpha_j}{x_j} + \beta_j x_j \right) + \sum_{j=1}^{n} \gamma_j \left[\left(\sum_{m=1}^{j} x_m \right)^{3/2} - \left(\sum_{m=1}^{j-1} x_m \right)^{3/2} \right] / x_j \, .$$

Here the first term is the fixed cost per unit, the second one the storage cost per unit and the last one the material handling cost per unit. In the last expression the ratios are not convex-concave, in fact they are not even quasiconvex. However, it can be shown that there exists a one-to-one continuous tranformation of variables such that the transformed function of $K(x)$ is (strictly) convex. This proves that a stationary point $\bar{x}$ $(\nabla K(\bar{x}) = 0)$ is a (unique) global minimum of $K(x)$. Hence the minimum of $K(x)$ can be calculated in a straightforward manner.

We see from this application that a sum-of-ratios problem (1.1) may well be tractable if the ratios are <u>not</u> concave-convex. Of course, the major challenge remains to derive properties for (1.1) when the ratios <u>are</u> concave-convex.

2. Maximizing the Smallest of Several Ratios

Consider

$$(1.3) \qquad \min_{1 \le i \le p} \frac{f_i(x)}{g_i(x)} \to \text{Sup}$$

subject to

$$h_k(x) \le 0 \qquad k = 1, \ldots, m$$

$$x \in S_0$$

where $S_0 \subseteq R^n$ is nonempty, convex

$h_k: S_0 \to R$ are convex.

In contrast to (1.1) problem (1.3) is known to have very useful properties in case of concave-convex ratios f_i/g_i. Before going into details we want to first point out some applications of (1.3).

In mathematical economics problem (1.3) arises when determining the growthrate of an expanding economy [26, 27]:

$$(1.4) \qquad \text{growthrate} = \max_x \left(\min_{1 \le i \le p} \frac{\text{output}_i(x)}{\text{input}_i(x)} \right)$$

where x denotes a feasible production plan of the economy.

In management science simultaneous maximization of rates can be achieved by either using a worst-case approach

$$(1.5) \qquad \min_{1 \le i < p} \frac{f_i(x)}{g_i(x)} \to \sup$$

or with help of prescribed ratio goals r_i

$$(1.6) \qquad \min_{1 \le i \le p} \left| \frac{f_i(x)}{g_i(x)} - r_i \right| \to \inf.$$

In both cases a max-min fractional program (1.3) is to be solved. The second approach is used in a financial planning model by Ashton/Atkins [1] where liquidity, return on capital, dividend per share as well as other financial ratios are maximized simultaneously. Model (1.6) is also used in a resource allocation problem by Charnes/Cox/Lane [6] designed to allocate funds under equity considerations to educational institutions in the state of Texas.

A third area of application of model (1.3) is numerical mathematics. Given the values F_i of a function $F(t)$ in finity many points t_i of an interval for which an approximating ratio of two polynomials $N(t; x_1)$ and $D(t; x_2)$ with coefficient vectors x_1, x_2 is sought. If the best approximation is defined in the sense of the L_∞-norm, then the following problem is to be solved:

$$(1.7) \qquad \max_i \left| \frac{N(t_i; x_1)}{D(t_i; x_2)} - F_i \right| \to \inf$$

for variables x_1, x_2 [3]. Like (1.6) this problem can be reduced to the max-min fractional program (1.3).

Several authors including J. von Neumann [26, 27] have introduced dual programs for problems of the type (1.3) using different approaches; for references see [8]. Recently two new approaches were suggested in [8] and [14]. The first one employs the framework of quasiconvex duality whereas the second one does not leave the confines of convex analysis. Very recently Scott/Jefferson [25] derived similar results with help

of conjugate duality.

In most duality approaches for (1.3) the following assumptions are made:

S_0 is compact

$-f_i$, g_i, h_k are lower-semicontinuous

$-f_i$, g_i are convex

g_i are positive on S_0

f_i are nonnegative on S, if at least one g_i is not affine.

Herein $S = \{x \in S_0 \mid h_k(x) \leq 0 \quad k = 1, \ldots, m\}$ denotes the feasible region which is assumed to be nonempty.

Let $F = (f_1, \ldots, f_p)^T$, $G = (g_1, \ldots, g_p)^T$ and $h = (h_1, \ldots, h_m)^T$. The following dual was derived in [14]:

$$(1.8) \qquad \sup_{x \in S_0} \frac{u^T F(x) - v^T h(x)}{u^T G(x)} \to \inf$$

subject to

$$u \geq 0, \; u \neq 0, \; v \geq 0.$$

Under the assumptions above one can show [14]: the optimal values in (1.3) and (1.8) are the same. Additional duality relations resembling those in convex and linear programming can be derived as well [8, 14].

We note that the primal max-min problem (1.3) is associated with a dual min-max problem (1.8). Such a symmetry is not obvious in single-ratio fractional programming duality theory [16]. Symmetry between the primal and dual exists also in the following sense: in both problems a local optimum is a global optimum. This follows from the fact that the primal objective function is semistrictly quasiconcave and the dual objective function is semistrictly quasiconvex [2].

The dual objective function usually involves infinitely many ratios in contrast to the primal one. However this asymmetry disappears in case of a linear problem (1.3) where f_i, g_i and h_k are affine and S_0 is the (unbounded) nonnegative orthant of R^n. Then only finitely many ratios need to be considered in the dual objective function [8, 14]. In the linear case it can further be shown that in addition to the usual complementary slackness between variables in one problem and constraints in the other one complementary slackness also exists between certain variables in one and ratios in the other one. For details see [8]. Summarizing, we can say that for linear problems (1.3) there exists complete symmetry as well as intimate relations between the primal and the dual fractional program.

We realize that for max-min fractional programs (1.3) with concave-convex ratios a rich duality theory can be established. We will now turn to an algorithm that solves these same fractional programs [9]. In the linear case of (1.3) the method can also be applied to the dual as well because of the symmetry between the primal and dual. The procedure is a generalization of Dinkelbach's algorithm [11] for single-ratio fractional programs. In case of one ratio only the method is theoretically as well as computa-

tionally very appealing [17, 22].

Consider

$$(1.9) \qquad \bar{\theta} = \max_{x \in S} \left(\min_{1 \le i \le p} \frac{f_i(x)}{g_i(x)} \right) \qquad (P)$$

where $S \subseteq R^n$ is nonempty, compact

f_i, g_i: $S \to R$ are continuous and $g_i > 0$.

We associate with (1.9) the parametric program

$$(1.10) \qquad F(q) = \max_{x \in S} \min_{1 \le i \le p} [f_i(x) - q\, g_i(x)], \quad q \in R. \qquad (P_q)$$

$F(q)$ is a strictly decreasing, continuous function. The idea of the algorithm is to find a zero $\bar{q}$ of F. It is easy to see that an optimal solution of $(P_{\bar{q}})$ is an optimal solution of (P), and $\bar{q} = \bar{\theta}$.

(P_q) is a convex program for $q \ge 0$, if f_i/g_i are concave-convex ratios. (P_q) can be written as a linear program in the linear case of (P), i.e. if f_i, g_i are affine and S is a convex polyhedron. In either case the parametric problem (P_q) (q given) is easier to solve than the original problem (P).

The following algorithm was suggested in [9]:

Step 1: Let $x^0 \in S$, $q_1 = \min_i \dfrac{f_i(x^0)}{g_i(x^0)}$, $k = 1$.

Step 2: Solve (P_{q_k}); let x^k be an optimal solution.

Step 3: If $F(q_k) = 0$, then stop; x^k is an optimal solution of (P) and $q_k = \bar{q} = \bar{\theta}$.

If $F(q_k) \neq 0$, let $q_{k+1} = \min_i \dfrac{f_i(x^k)}{g_i(x^k)}$, $k = k + 1$,

and go back to Step 2.

The sequence $\{q_k\}$ is increasing and converges to $\bar{q}$. It does so linearly. Furthermore, each convergent subsequence of $\{x_k\}$ converges to an optimal solution of (P). For details see [9].

It is interesting to contrast the algorithm by Dinkelbach [11] for $p = 1$ with its generalization for $p > 1$ in [9]. For $p = 1$ we have:

a) F is convex on R,
b) the algorithm is equivalent to Newton's algorithm, i.e. it generates the same sequence $\{q_k\}$, and
c) the algorithm converges superlinearly.

For $p > 1$ none of these properties hold anymore. In particular, the algorithm is only linearly convergent in general. On the other hand, the procedure has two advantages over Newton's method:

1) F need not be convex, and

2) a subgradient of F need not be known.

The disadvantage of slow convergence can be overcome in cases when (P) has a unique optimal solution. This is achieved by solving a somewhat modified subproblem in Step 2 [10, 13]:

$$(1.11) \qquad \max_{x \in S} \left(\min_{1 \le i \le p} \frac{1}{g_i(\tilde{x}^{k-1})} \left[f_i(x) - \tilde{q}_k \, g_i(x) \right] \right) .$$

It can be shown that $\{\tilde{q}_k\}$ is increasing and converging to $\bar{q}$. It does so linearly. However, if the optimal solution of (P) is unique, then $\{\tilde{q}_k\}$ converges even superlinearly and $\{\tilde{x}^k\}$ converges to an optimal solution of (P). For details see [10]. Under more restrictive assumptions on (P) the convergence rate can be made 1.6 or even quadratic [5, 13].

The computational experience with the original algorithm in [9] and its modification in [10] is presented in [12]. The modified algorithm proves to be clearly superior to the original one.

So far the feasible region of (P) was assumed to be bounded. In the unbounded case $F(q)$ may no longer be finite and then the algorithm will fail to work. However, it can be shown [9] that at least in the linear case of (P) the dual as introduced earlier will always have a finite F-function. The algorithm can therefore be applied to the dual. Hence the problem of an unbounded feasible region of (P) can be overcome by solving the dual instead of the primal, at least in the linear case of (P).

3. Multiobjective Fractional Program

Consider the general multiobjective program

$$(1.12) \qquad \max_{x \in S} \left(F_1(x), \ldots, F_p(x) \right) , \qquad p \ge 2$$

where $S \subseteq R^n$ is nonempty, compact,

F_i: $S \to R$ are continuous.

Let E be the set of efficient (Pareto-optimal) solutions and $F(E)$ the efficient frontier.

Bitran/Magnanti [4] showed that E is a connected set, if S is convex and all F_i are concave functions. We recall that a set M is called connected if there do not exist closed sets Q_1, Q_2, such that $M \subseteq Q_1 \cup Q_2$, $M \cap Q_1 \neq \emptyset$, $M \cap Q_2 \neq \emptyset$ and $M \cap (Q_1 \cap Q_2) = \emptyset$.

The question arises for which nonconcave functions F_i the set E is still connected. In particular, will it be connected for multiobjective fractional programs, and for which kind? Unfortunately, it does not seem to be possible to carry over the analysis by Bitran/Magnanti to nonconvex multiobjective programs since the authors use the parametrization $\sum_i \lambda_i F_i(x)$ to generate E.

In case of a single objective function ($p = 1$) E is identical with the set of optimal solutions. This set is still convex, and thus connected, if $F(x)$ is quasiconcave [2].

In the bicriteria case ($p = 2$) quasiconcavity of F_1, F_2 does not always guarantee connectedness of E as the following example shows:

$$S = [-1, 1],\ F_1(x) = x,\ F_2(x) = \begin{cases} -x & -1 \leq x \leq 0 \\ 0 & 0 \leq x \leq 1 \end{cases}.$$

Here $E = [-1, 0) \cup \{1\}$. This example also demonstrates why E is disconnected; namely because of the flat portion of $F_2(x)$. On the other hand, if F_1, F_2 are both semistrictly quasiconcave [2] (and not just quasiconcave), then E can be shown to be connected [20]. The proof makes use of the parametrization

$$(1.13) \qquad \begin{aligned} &\max_{x \in S} F_2(x), \\ &F_1(x) \geq \alpha \end{aligned} \qquad \underline{F}_1 \leq \alpha \leq \bar{F}_1 .$$

E is the union of the sets of optimal solutions of (1.13) each of which is convex and therefore connected.

Now let us consider the special case of a bicriteria fractional program

$$(1.14) \qquad \max_{x \in S} \left(\frac{f_1(x)}{g_1(x)}, \frac{f_2(x)}{g_2(x)} \right).$$

If f_i/g_i are concave-convex, then these functions are semistrictly quasiconcave [2]. Hence E is connected for bicriteria fractional programs where both ratios satisfy the classical assumption. E can be calculated by solving a one-parametric sequence of single-ratio fractional programs.

$$(1.15) \qquad \begin{aligned} &\max_{x \in S} \frac{f_2(x)}{g_2(x)}, \\ &\frac{f_1(x)}{g_1(x)} \geq \alpha \end{aligned}$$

As shown in [15, 16], (1.15) can be reduced to a convex program with help of a variable-transformation. The bounds $\underline{F}_1$, $\bar{F}_1$ are calculated by solving two additional convex programs. Hence the set E of a bicriteria fractional program with concave-convex ratios can be determined with help of convex programming alone [20].

Once E is known, the "hard" problem in Section 1

$$(1.16) \qquad \max_{x \in S} \left(\frac{f_1(x)}{g_1(x)} + \frac{f_2(x)}{g_2(x)} \right)$$

can be solved by finding an optimal solution of

$$(1.17) \qquad \max_{x \in E} \left(\frac{f_1(x)}{g_1(x)} + \frac{f_2(x)}{g_2(x)} \right)$$

with a reduced feasible region E instead of S. This is so since one of the optimal solutions of (1.16) is an efficient solution of (1.14).

In case of three criteria the following result was recently derived [7]: If F_1, F_2, F_3 are semistrictly quasiconcave, then the closure of $F(E)$ ($cl\, F(E)$) is connected. The proof uses the parametrization

$$(1.18) \qquad \max_{x \in S} (F_2(x), F_3(x))$$

$$F_1(x) \geq \alpha$$

and takes advantage of the connectedness of the set of efficient solutions in the bicriteria case. As a special case the multiobjective fractional program

$$(1.19) \qquad \max_{x \in S} \left(\frac{f_1(x)}{g_1(x)}, \frac{f_2(x)}{g_2(x)}, \frac{f_3(x)}{g_3(x)} \right)$$

with concave-convex ratios f_i/g_i may be considered.

While $F(E)$ is closed for $p = 2$, this is not true for $p = 3$ anymore. Hence $cl\, F(E) \neq F(E)$. However if it is true that not only $cl\, F(E)$ but even $F(E)$ itself is connected, then also E is connected because of quasiconcavity of F_i.

At the present time two open questions remain:

1. Is even $F(E)$ (and therefore E) connected for $p = 3$?
2. Is E connected for $p > 3$?

Based on the experience with $p \leq 3$ semistrict quasiconcavity of F_i has to be assumed when answering these questions. This included the case of concave-convex ratios.

References

1. Ashton, D.J. and D.R. Atkins, "Multicriteria Programming for Financial Planning," Journal of the Operational Research Society 30, 1979, 259-270.

2. Avriel, M., Diewert, W.E., Schaible, S. and I. Zang, Generalized Concavity, Plenum Publishing Corp., New York, 1987.

3. Barrodale, I., "Best Rational Approximation and Strict Quasiconvexity," SIAM Journal of Numerical Analysis 10, 1973, 8-12.

4. Bitran, G.R. and T.L. Magnanti, "The Structure of Admissable Points with Respect to Cone Dominance," Journal of Optimization Theory and Applications 25, 1979, 573-614.

5. Borde, J. and J.-P. Crouzeix, "Convergence of a Dinkelbach-type Algorithm in Generalized Fractional Programming," Zeitschrift fur Operations Research 31, 1987, A31-A54.

6. Charnes, A., Cox, L. and M. Lane, "A Note on the Redesigning of a Rate Structure for Allocation of State Funds to Educational Institutions," Working Paper 70-49 of Project GUM, University of Texas at Austin, May 1970.

7. Choo, E.U., Schaible, S. and K.P. Chew, "Connectedness of the Efficient Set in Three Criteria Quasiconcave Programming," Cahiers du Centre d'Etudes de Recherche Operationelle 27, 1985, 213-220.

8. Crouzeix, J.-P., Ferland, J.A. and S. Schaible, "Duality in Generalized Linear Fractional Programming," Mathematical Programming 27, 1983, 342-354.

9. Crouzeix, J.-P., Ferland, J.A. and S. Schaible, "An Algorithm for Generalized Fractional Programs," Journal of Optimization Theory and Applications 47, 1985, 35-49.

10. Crouzeix, J.-P., Ferland, J.A. and S. Schaible, "A Note on an Algorithm for Generalized Fractional Programs," Journal of Optimization Theory and Applications 50, 1986, 183-187.

11. Dinkelbach, W., "On Nonlinear Fractional Programming," Management Science 13, 1967, 492-498.

12. Ferland, J.A. and J.Y. Potvin, "Generalized Fractional Programming: Algorithms and Numerical Experimentation," European Journal of Operational Research 20, 1985, 92-101.

13. Flachs, J., "Generalized Cheney-Loeb-Dinkelbach-Type Algorithms," Mathematics of Operations Research 10, 1985, 674-687.

14. Jagannathan, R. and S. Schaible, "Duality in Generalized Fractional Programming via Farkas' Lemma," Journal of Optimization Theory and Applications 41, 1983, 417-424.

15. Schaible, S., "Parameter-free Convex Equivalent and Dual Programs of Fractional Programming Problems," Zeitschrift fur Operations Research 18, 1974, 187-196.

16. Schaible, S., "Duality in Fractional Programming — A Unified Approach," Operations Research 24, 1976, 452-461.

17. Schaible, S., "Fractional Programming, II. On Dinkelbach's Algorithm," Management Science 22, 1976, 868-873.

18. Schaible, S., "A Note on the Sum of a Linear and Linear-Fractional Function," Naval Research Logistics Quarterly 24, 1977, 691-693.

19. Schaible, S., Analyse und Anwendungen von Quotientenprogrammen -- ein Beitrag zur Planung mit Hilfe der nichtlinearen Programmierung, Hain-Verlag, Meisenheim, 1978.

20. Schaible, S., "Bicriteria Quasiconcave Programs," Cahiers du Centre d'Etudes de Recherche Operationelle 25, 1983, 93-101.

21. Schaible, S., "Simultaneous Optimization of Absolute and Relative Terms," Zeitschrift fur Angewandte Mathematik und Mechanik 64, 1984, 363-364.

22. Schaible, S. and T. Ibaraki, "Invited review: Fractional Programming,"European Journal of Operations Research 12, 1983, 325-338.

23. Schaible, S. and T.J. Lowe, "A Note on a Material Control Problem," AIIE Transactions 15, 1983, 177-179.

24. Schaible, S. and W.T. Ziemba (eds.), Generalized Concavity in Optimization and Economics, Academic Press, New York 1981.

25. Scott, C.H. and T.R. Jefferson, Conjugate Duality in Generalized Fractional Programming, Graduate School of Management, University of California at Irvine, March 1987.

26. von Neumann, J., "Uber ein ökonomisches Gleichgewichtssystem und eine Verallgemeinerung des Brouwerschen Fixpunktsatzes," Ergebnisse eines mathematischen Kolloquiums, Hrsg. v. K. Menger, Heft 8, Leipzig-Wien 1937, 73-83.

27. von Neumann, J., "A Model of General Economic Equilibrium," Review of Economic Studies 13, 1945, 1-9.

II. Optimization and Parallel Processing

AN EXPERIENCE OF ADVANCED COMPUTATION TECHNIQUES IN THE SOLUTION OF NONLINEARLY CONSTRAINED OPTIMIZATION PROBLEMS

D. Conforti , L. Grandinetti
Dipartimento di Sistemi - Università della Calabria
Cosenza - Italy

ABSTRACT

Vector and parallel processing are advanced computation techniques which allow to realistically solve a number of problems traditionally intractable.

In this paper a short review of some basic concepts of parallel computing is given; in addition, motivations are provided to use parallel algorithms for solving very difficult optimization problems; lastly, the main progress obtained in the parallel nonlinear optimization methods is roughly sketched.

A large scale problem of great practical interest, related to the production, transmission and distribution of electric energy in a network has, then, been taken into consideration and the optimal management policy for the system has been formulated as a nonlinear mathematical program. Its solution by a sequential implementation of a gradient-type algorithm exhibits a very low efficiency; thus, the vectorized implementation of a variable metric algorithm is considered here. The numerical results show its remarkable effectiveness and speed; therefore, the evidence is provided that the optimal operation of a power electricity network, even very large, can be successfully computed by a vector supercomputer when a sound and appropriate nonlinear optimization algorithm is used.

1. INTRODUCTION

One of the main difficulties in dealing with large scale, complex computational problems, depends on the use of conventional (serial) computers, which limit -often in a severe way - an effective and fast solution. In very recent years the availability of vector and parallel computers (e.g. CRAY I, CRAY X MP, CRAY 2; CDC CYBER 205, ICL DAP, Distributed Systems, etc.) has made possible, for their own particular architecture, the effective solution of many difficult problems.

In the very near future, the advent of new parallel machines, at present available as prototypes, and in particular of compact minisupercomputers, will allow a very general application and an "at hand" availability of them for complex numerical computations related to mathematical problems and models in many theoretical and applied fields.

In this paper some basic concepts of vector and parallel computing are preliminarly reviewed; then, a motivation is given - together with some references- for applying the new advanced

computation techniques to the solution of large scale nonlinear optimization problems. In particular, the problem of optimal management of an electric power network, under the respect of its economic operation, has been taken into consideration.

An electric power network is a complex connection of several elements (power generators, trasformers, transmission lines, shunt capacitors and inductors, loads, etc.), whose fundamental goal is that of generating, transmitting and distributing electric energy, such that the demand for electric energy is fully satisfied.

Since this important aim, optimization problems need, very often, to be solved for properly planning and managing the electric power systems, in such a way that the electric energy be supplied in the most economic, reliable and secure way.

These problems, called Optimal Power Flow problems (OPF), are typically formulated as nonlinear programming problems, involving both equality and inequality constraints, and are characterized by a large number of variables and constraints, so that they are very hard to be solved by using classical algorithms.

Although many of them have been proposed and applied, nevertheless the development of new algorithms particularly suitable for large scale systems, i. e. characterized by increased efficiency and robustness, is of very crucial importance and still an open objective of research.Therefore the use of parallel processing in the solution of OPF problems appears attractive and desirable, providing a non conventional powerful tool for overcoming the inherent limitations of sequential algorithms and thus for gaining extrahigh efficiency; in addition it is now widely possible, without severe difficulties to have access at least to vector supercomputers.

A gradient type algorithm, namely the version of the Sequential Gradient Restoration Algorithm (SGRA) recently proposed by Miele, Sims, Basapur and Heideman [1] , was firstly used, in its sequential implementation, for the solution of OPF problems; recently it was also implemented on a vector computer CDC Cyber 205, and numerically tested on the same problems [2].

The numerical results, show that the SGRA algorithm, if properly used, may be useful in dealing with the solution of OPF problems, for many practical aspects.

Nevertheless, a different algorithm based on a sound variable metric method is proposed in this paper; it seems preferable not only theoretically but also computationally.

Finally, the deep superiority - in terms of efficiency - of the parallel implementation versus the sequential one, is pointed out and some particular features are commented.

2. VECTOR AND PARALLEL COMPUTING

The development of automatic computation, by using computing systems, has always been characterized by limitations and difficulties. Among these one of the most important is the computing speed, that is to say the capacity of the system to solve, in a reasonable time and in a reliable way, a very complex problem.

Then, mostly with respect to the solution of large scale scientific problems, computing systems

characterized by powerful hardware and software resources have always been required.

In the past years, the increase of computing speed has been obtained by more and more sophisticated electronic technology. But nowadays it is really difficult to design new and more powerful computing systems based only on technological improvements; to increase the performance it is necessary to take into account the overall computer architecture, i. e. to design new organization models of computing systems.

At present the information processing is dominated by the "sequential monoprocessing" methods, whose peculiar characteristics depend, directly, on the architectural organization of Von Neumann computer (the so called "control flow machine").

In fact, in a conventional computer, the operations (e. g. memory fetch or store, arithmetic or logical operations, input or output operations) can be executed only in a sequential fashion, according to the order fixed in advance by the program; further, in the same time instant, it is impossible to execute more than one operation.

To overcome the inherent limits due to the architecture of classical computer systems, it may be very convenient to use computers based on non - Von Neumann architecture, realized according the principles of parallel processing.

Parallel processing is an efficient form of information processing which emphasizes the exploitation of "concurrent activities" in the computing processes. In other words, the concept of parallelism refers to the ability to overlap or perform simultaneously many concurrent activities.

These concurrent activities are attainable in a computing system at various processing levels: job or program level, task or procedure level, interinstruction level, intrainstruction level.

The most important ways of introducing parallelism into the architecture of computers may be summarized as:

i) pipeline processing: the arithmetic/logic unit (ALU) and the control unit (CU) are organized as in a typical assembly-line; more precisely, their organization is such that each operation is subdivided in a number of steps, each step consisting in the execution of a suboperation. The steps relative to the execution of a given operation are suitably chained. In this way a number of operations can be, partially, overlapped thus exploiting a "temporal" parallelism; consequently the ALU and CU performance can be remarkably improved;

ii) array processing : an array processor is a synchronous parallel computer with multiple arithmetic-logic units, called processing elements, under common control, that can operate in parallel in a lockstep fashion;

iii) multiprocessing : several independent programs or tasks can be performed by a multiprocessor architecture containing two or more processor units, which share access to common sets of memory modules, I/O channels and peripheral devices.

Pipeline and array processors are very suitable for vector computation, where component operations are data independent and may be repeated many times. In fact an array computer is able to perform an operation simultaneously on a large number of different data, while, due to overlapped

instruction and arithmetic execution, it is obvious that a pipeline machine is better tuned to perform the same operations repeatedly through the pipeline.

A vector-pipelined computer has a parallel structure which, roughly speaking, deals with a vector operand essentially in the same way in which a serial machine deals with a scalar operand. This is possible since the general architectural model of a vector computer is based on several and independent functional units - of pipeline type -, each of which is dedicated to execute a specific operation (addition, multiplication, division, square root, logical operations, etc.) and can operate on different data in a parallel way. Moreover the central memory is organized in several independent memory banks, by which it is possible to execute contemporary access.

The software production for this type of machine is characterized by the use of vector high level language, in particular extended FORTRAN language, characterized by "vector" facilities, which allow very efficient operations on vectors and matrices.

However vector processing represents a low level of parallelism, because the concurrent activities are exploited at level of vector operations, whereas, as far as the machine architecture is concerned, there is still a strictly connection with the structure of Von Neumann computer.

An high level parallelism is attainable by multiprocessors and distributed systems, whereby concurrent activities can be exploited and executed at task or procedure level.

Concerning some of the most important aspects of future parallel processing, it is worth while to mention the effort currently done to develop parallel programming styles based on "non-procedural" models of computation, such as functional programming, data flow computation, logic programming and object-oriented programming. In fact, the present parallel computers, even if show new architectural aspects, are equipped by software tools still of "procedural" type and realized on purpose for Von Neumann computational model.

3. PARALLEL NONLINEAR OPTIMIZATION

The availability of parallel computers represents a good opportunity in order to improve the solution methods of non linear optimization problems.

At present a great number of optimization algorithms exists, based on excellent theoretical framework and for which satisfactory computational results have been obtained when using sequential machines on reasonably complex problems.

However it is very important to point out that the numerical techniques for solving non standard problems are strongly insufficient. In fact, mainly in the field of applications, there are problems, whose solutions are sought via conventional computers, which can cause severe computational difficulties; some of these are related to the following aspects:

a) computing speed not sufficiently high when a real time or a global solution is needed;

b) excessive memory locations keeping, and rounding errors propagation, typically due to the large scale dimension of the problem (thousands of variables and several thousands of

constraints) which may also cause an intolerable time consumption.

In this context, parallel processing methodologies can bring considerable advantage, allowing the implementation of more complex and sophisticated algorithms which may exploit more powerful and original theoretical conceptions.

Unfortunately in the past two decades there has been a limited amount of experience with nonlinear optimization algorithms on parallel computers.

Only in the field of unconstrained optimization some authors proposed parallel variants of classical sequential algorithms, in a period long before parallel computers became commercially available.

Among the initial attempts, Avriel and Wilde [3] as well as Karp and Miranker [4] developed a parallel Fibonacci algorithm for the linear search; Chazan and Miranker [5] proposed a conjugate direction method is which geometrically parallel directions are explored by parallel linear searches.

Pierre [6] studied the Newton-Raphson method using the parallel computing facilities to generate finite-difference approximations to the first and second derivatives. Straeter [7] was the first author who proposed parallel variable-metric algorithm; his idea has recently been generalized by Van Laarhoven [8] . Finally, it is worth mentioning that Straeter and Markos [9] as well as Kowalik and Ramakrishnan [10] described a parallel version of the Jacobson-Oksman [11] method based on the so-called homogeneous model function.

In very recent years a considerable amount of numerical experiments on parallel computers (in particular ICL-DAP with 4096 processing elements), have been carried out by Dixon, Patel et al. [12], [13], [14], [15]. They developed, essentially, the idea of replacing the traditional one-dimensional searches by multi-dimensional searches, exploiting the parallelism at level of simultaneous evaluation of the objective function and its gradient at various points.

Concerning the application of parallelism in order to solve nonlinear constrained optimization problems, Dayde [16] has recently carried out a work in which he solves a particular class of primal and dual structural-design problems by using the parallel uncontrained optimization methods of Straeter [7] and Jacobson-Oksman [11]. The solution of large scale optimization problems via decomposition and parallel computing is currently under investigation by De Silva et al. [17].

Finally, Schnabel [18] and Lootsma and Ragsdell [19] proposed two interesting general surveys about parallelism in optimization.

In spite of their interesting attempts, the above mentioned research works have not yielded substantial contributions in order to exploit the real potentialities of the parallel processing applied to the solution of nonlinear optimization problems.

This partially depends on limited production of computational testing, which is, principally, ascribed to the short availability of parallel machines.

Another serious complication is due to the fact that there are several architectural models of parallel computer, each with its own particular advantages and disadvantages. Thus, to detect the

best parallel algorithm for a given problem, according to a performance criterion, becomes very difficult, because it depends on the particular architectural model used.

Sometimes, the parallelism has, essentially, been exploited at level of simultaneous evaluation, at several points, of the objective function, its gradient and, in some cases, constraint functions; this represents a low level of parallelism without changing, in general, the mathematical structure of algorithms.

A more deep mathematical analysis of sequential algorithms, together with a proper computational testing, can give a remarkable contribution to the design of parallel algorithms. However, to obtain a substancial progress in the parallel nonlinear optimization it is still necessary to develop the research activity in several directions. One of these directions concernes special cases and special structures, for instance large scale problems with interesting structural properties, which can be, efficiently, exploited by using parallel processing. But, only when a suitable "programming environment" for parallel processing will be available, it will be likely to realize a drastic "jump of quality" in the parallel optimization computations.

4. THE OPTIMAL POWER FLOW PROBLEM

An electric power system, aimed to fully satisfy, at any time, the electric energy demand, must achieve the fundamental goal of generating, transmitting and distributing electric energy, pursuing as priority objectives the economy, the reliability and the security of the electric service.

To meet these extremely important requirements, the system planning and management activities must be carried out in a way not only consistent and appropriate but also suitable to achieve the best performance. The formulation of a mathematical model of the system is then the basis for a sound and effective decision support; it, also, allows to naturally devise an optimization problem in which decision variables are, generally, bounded by limits of technical and management type, of security and energy balance.

These problems, known as Optimal Power Flow (OPF) problems, are formulated as nonlinear programs, involving both equality and inequality constraints, of the following general form:

$$\min \; f(x), \; x \in \mathbb{R}^n$$

s.t.

$$h(x) = 0, \quad h : \mathbb{R}^n \longrightarrow \mathbb{R}^m \;, \quad m < n$$

$$g(x) \leq 0, \quad g : \mathbb{R}^n \longrightarrow \mathbb{R}^p,$$

where:

$x \in \mathbb{R}^n$: vector of electric quantities, which are the decision variables;

$f(x)$: objective function; it assumes several meaning according to the goals to pursue;

$h(x) = 0$: network equations (load flow equations); they represent the energy balance in electric network for a given configuration of the network;

$g(x) \leq 0$: technical, management and security constraints on the main electric quantities involved in the network.

The explicit formulation of an optimal power flow problem, arising from a power system of great practical importance, is given in the sequel. For this purpose the electric power system is modelled as a network made up by nodes, which represent energy sources or loads, interconnected by branches which are models of the transmission lines.

The number of network's nodes and the number of lines are denoted by:

N = total number of nodes,

N_g = number of nodes where the active power is generated,

N'_g = number of nodes where the reactive power is generated,

N_L = number of lines.

The nodes are characterized by the following parameters:

P_g = N_g- dimensional vector representing the active power generated,

Q_g = N'_g - dimensional vector representing the reactive power generated,

P_d = N - dimensional vector representing the active power demanded,

Q_d = N - dimensional vector representing the reactive power demanded,

V = N - dimensional vector representing the magnitude of the voltage at the generic node.

δ = (N-1) - dimensional vector representing the phase angle of the voltage at the generic node. It is assumed that the voltage phase of the N-th node is given, for example $\delta_N = 0$.

Y = (N x N) - dimensional symmetric matrix representing the magnitude of the nodal admittances.

θ = (N x N) - dimensional symmetric matrix representing the argument of the nodal admittances.

The economic aspect of the network operation is of fundamental interest in our context. From this point of view it comes out, essentially on experimental basis, that a quadratic model for the cost of the entire active power generation is appropriate. To be precise, this cost, namely C, can be satisfactorily represented in the following form:

$$C = P^T_g \, D(a_i) \, P_g + b^T P_g + \Sigma \, e^T_i \, c, \quad i = 1, \ldots, N_g, \tag{1}$$

where:

$D(a_i) = \text{diag}(a_i)$, e_i = i-th vector of the canonical basis in $\mathbb{R}^{N_g}$.

Moreover the parameters a, b, c, which are vectors of dimension N_g, depend on the specific network considered and can be obtained by appropriate measurement.

However, it has to be taken into account that the power generated must be such as to satisfy the following constraints:

$$GP_g - P_d - P(V, \delta) = 0 \tag{2}$$

$$G'Q_g - Q_d - Q(V, \delta) = 0. \tag{3}$$

In these equations, which represent the energy balance in the network, G is a $N \times N_g$ matrix, whose element g_{ij} is equal to one if the i-th node in the network is the j-th generation node, otherwise is equal to zero; analogously, the $(N \times N'_g)$ matrix G' plays the same role for the reactive generation.

In (2) and (3) the terms $P(V, \delta)$, $Q(V, \delta)$ represent the power lost in the network; their generic components have the following expression:

$$P_i = V_i \Sigma_k V_k Y_{ik} \cos(\delta_i - \delta_k - \theta_{ik}), \quad i = 1, \ldots, N \tag{4}$$

$$Q_i = V_i \Sigma_k V_k Y_{ik} \cos(\delta_i - \delta_k - \theta_{ik}), \quad i = 1, \ldots, N \tag{5}$$

where the sum Σ_k is extended to all nodes k directly connected to the generic node i.

The following constraints must also be satisfied:

$$P^m_g \leq P_g \leq P^M_g \tag{6}$$

$$Q^m_g \leq Q_g \leq Q^M_g \tag{7}$$

$$V^m \leq V \leq V^M \tag{8}$$

$$\delta^m \leq \delta \leq \delta^M \tag{9}$$

The previous inequalities are nothing but technical constraints on the main electric quantities involved in the network and depend on its intended purpose.

For a network of general type, it is also necessary to consider the following "security" constraints:

$$S(P_g, Q_g) \leq S^M \tag{10}$$

where $S(P_g, Q_g)$ is the N_g-vector whose generic component has the following expression:

$$P_{g_i}^T P_{g_i} + Q_{g_i}^T Q_{g_i} \leq S^M_i, \quad i = 1, \ldots, N_g, \tag{11}$$

and

$$I\,(V,\,\delta\,) \le I^{M}\,. \tag{12}$$

It is worth noting that the first member of (9) represents the square of the apparent power globally generated in the network; it is natural that the apparent power be upper bounded by S^{M}.

Moreover, the current in every branch must be also limited, as indicated by the constraint (12). The quantity $I\,(V,\,\delta)$ is the N_L-vector whose generic component has the following expression:

$$I_i(V,\,\delta) = [V^2_j + V^2_k - 2V_jV_k \cos(\,\delta_j - \delta_k)] \,/\, Z^2_i\,; \tag{13}$$

the meaning of the scalar $I_i(V,\,\delta\,)$ is the squared modulus of the current in the i-th branch whose impedance is Z_i.

Finally it is useful to introduce a limitation on the range of values covered by the voltage phase displacement between two adjacent nodes. If $\delta(i)$ and $\delta(j)$ denote the phase of voltage at the adjacent nodes i and j, then this limitation can be written as follows:

$$|\,\delta(i) - \delta(j)\,| \le \Delta^{M}\,, \tag{14}$$

where Δ^{M} is a N_L vector representing the upper bound on the phase displacement.

Since, for the sake of economic reasons, the important target is to minimize the power generation cost, accordingly to the network constraints, it is possible to formulate the following nonlinear optimization problem:

$$\min \{\, P^{T}_g\, D(a_i)\, P_g + b^{T}\, P_g + \Sigma_i\, e^{T}_i\, c \,\} \tag{15}$$

s.t.

$$GP_g - P_d - P(V,\,\delta\,) = 0$$

$$G'Q_g - Q_d - Q(V,\,\delta\,) = 0$$

$$V^{m} \le V \le V^{M}$$

$$P^{m}_g \le P_g \le P^{M}_g$$

$$Q^{m}_g \le Q_g \le Q^{M}_g$$

$$\delta^{m} \le \delta \le \delta^{M}$$

$$S(P_g,\,Q_g) \le S^{M}$$

$$I\,(V,\,\delta\,) \le \delta^{M}$$

$$|\,\delta(i) - \delta(j)\,| \le \Delta^{M}\,.$$

This nonlinear optimization model is characterized by several aspects, which represent a serious complication for its numerical solution:

- the constraints are highly nonlinear;
- the number of variables and the number of constraints, for a typical, significant, realistic application are very large (thousands and several thousands);
- the most number of constraints are inequality constraints.

Therefore, the particular structure of the OPF problem (specially the fact that the problem is highly nonlinear and constraints are mostly inequality constraints) suggests to use a method of solution possessing excellent theoretical properties (e. g. local and global convergence) particularly suitable in those difficult situations depending on the loads of the network.

5. THE VARIABLE METRIC METHOD

An early approach to the solution of problem (15) was based on a gradient-type method, the Sequential Gradient Restoration Algorithm, proposed by A. Miele et al. [1]. Although quite effective for small size problems and useful (all iterates belonging to the feasible region), nevertheless it proves too slow for successfully solve large scale problems in essentially real time. In addition, its convergence is not guaranteed in general situations.

As previously pointed out, many factors make the solution of OPF problems very difficult computationally. For this reason it has been considered advisable to use the Variable Metric Method (VMM) proposed by S. P. Han [20] and M. J. D. Powell [21], which has proved very sound theoretically and potentially capable of remarkable computational performances. A review of it is shortly provided here.

The constrained optimization problem:

$$\min F(x), \quad F : \mathbb{R}^n \longrightarrow \mathbb{R} \tag{16}$$

s.t.

$$c_i(x) = 0 \qquad i \in E$$

$$c_i(x) \geq 0 \qquad i \in I, \qquad c_i : \mathbb{R}^n \longrightarrow \mathbb{R}\ , \quad \forall\, i$$

is solved as a sequence of quadratic programming subproblems, based on function and gradient information at the current estimate x_k of the solution x^* . The generic subprogram is:

$$\min Q(d) \equiv F(x_k) + d^T \nabla F(x_k) + \frac{1}{2} d^T B_k d \tag{17}$$

s.t.

$$c_i(x_k) + d^T \nabla c_i(x_k) = 0\ , \qquad i \in E$$

$$c_i(x_k) + d^T \nabla c_i(x_k) \geq 0\ , \qquad i \in I\ .$$

Its solution defines, at any iteration k, the search direction d_k and then

$$x_{k+1} = x_k + \alpha_k d_k\ . \tag{18}$$

The steplength parameter $\alpha_k \in \mathbb{R}_+$, extremely important for forcing convergence of the sequence

$\{x_k\}$ to x^* , is provided by a suitable line search which:

(a) must reduce the objective function,

(b) must satisfy the constraints.

The matrix B_k in (17), intended to approximate, in quasi-Newton sense, the second derivative matrix of the Lagrangian function, involves the Lagrange multipliers of the quadratic subprogram, which approximate those of the original problem.

The choice of α_k is crucial. Among the different strategies on which the line search is based, an augmented Lagrangian type line search function has been considered here, according to Schittkowski [22] .

It is worth noting that the method is mathematically very sound, since global and Q-superlinear convergence properties are proved [23] .

6. VECTOR SUPERCOMPUTING

An inherent limitation in computational performance affects available optimization methods (including the variable metric method, in spite of its sound properties) when they are used on conventional serial computing systems. In fact when solving a nonlinear program for a network model represented by (15), the large (and also very large) dimension is a typical feature and also the real time solution is required. A "time horizon" of minutes, typical for getting the benefits of optimizing the economic dispatching of electric energy, makes this type of solution very crucial.

In these situations, very difficult to overcome with existing methods and standard sequential computational techniques, the parallel computing approach could be very useful to achieve a substantial improvement in performance.

The possibility to detect parallelism in the OPF problems depends, above all, on the topology of the electrical network, the structure of the optimization model and the algorithm used for solving the required problem.

As regards the topology of the network, some decomposition techniques (e.g. tearing operations on the network) may be applied to exploit parallelism. For each obtained subnetwork, an independent OPF subproblem can be solved, simultaneously with each other, to obtain a "partial" solution . On the basis of these solutions, a "whole" solution can be found, successively, taking into account the electrical conditions of tearing.

This type of parallelism can be, efficiently, implemented in a multiprocessors or multicomputers environment, with "master-slave" processors organization. The "slave" processors carry out, in parallel, the independent OPF subproblems, whereas the "master", once received the "partial" solutions from the "slaves", computes the "whole" solution of the entire network.

Although appealing, this naturally "parallel" approach may suffer for the present lack of concurrent algorithms conceived for successfull implementation on computer systems with highly parallel architecture.

However a lower type of parallelism can readily be exploited by using modern, commercially available, vector supercomputers.

In these machines the parallelism is low and is performed essentially at arithmetic level by exploiting the vector and matrix operations. In fact the peculiar characteristic of vector processing is the capacity to deal with a vector operand, i.e. an ordered set of N elements, where N is the length of the vector, in the same way in which sequential computation deals with a scalar operand.

To examine the efficiency of vector processing over scalar processing, it is worthwhile to compare the following two programs, one written for vector processing and the other written for scalar processing.

In a conventional scalar processor, the FORTRAN DO loop

```
      DO 1 I = 1, N
        A(I) = C(I) + D(I)
        B(I) = S + A(I)
1     CONTINUE
```

is implemented by the following sequence of scalar operations:

```
      INITIALIZE   I =1
1     LOAD C(I)
      LOAD D(I)
      ADD  C(I) + D(I)
      STORE A(I) ← C(I) + D(I)
      LOAD A(I)
      LOAD S
      ADD  S + A(I)
      STORE B(I) ← S + A(I)
      INCREMENT  I ← I + 1
      IF I ≤ N GO TO 1
      STOP
```

On a vector computer the above DO loop operations can be vectorized into two vector instructions in a sequence:

```
A(1; N) = C(1; N) + D(1; N)
B(1; N) = S + A(1; N)
```

where the notation A(1;N) refers to the N-element vector A(1), A(2), . . ., A(N).

It can be easily seen that the execution of the scalar loop repeats the loop-control overhead in each iteration. In vector processing using pipelines, the overhead is reduced by using hardware or firmware controls.

One of the fundamental motivations for implementing the VMM on a vector computer is based on

the consideration that the amount of computational work is strongly affected by the dimensions of vectors and matrices playing a decisive role in the execution process of the algorithm.

Moreover, the vector machine makes possible to code the algorithm without any drastic reformulation or change of its conceptual structure, overcoming the difficulties wich might arise in implementing, for instance, decomposition techniques in a multiprocessing environment.

A typical feature of vector processing consists in the matrix product execution. From a mathematical point of view, the product of two matrices A and B (N x M and M x N dimensions respectively) is a N x N matrix C, whose elements are related to A and B elements by the following expression:

$$C(I,J) = \sum_{k=1}^{M} \{ A(I,K) \times B(K,J) \}, \quad I,J = 1,N$$

In a conventional serial computer the above equation can be implemented using FORTRAN language, as follows:

```
      DO 3 I = 1, N
        DO 2 J = 1, N
          DO 1 K = 1, M
            C(I, J) = C(I, J) + A(I, K) x B(K, J)
1       CONTINUE
2     CONTINUE
3   CONTINUE
```

where it is assumed that all elements C(I, J) are set to zero before entering the code.

In this way each element of C is computed sequentially as scalar product between the rows of A and the columns of B.

In a vector computing environment it is possible to rearrange the code in such a way, for instance, to compute each column of C as a linear combination of columns of A having the elements of the respective column of B as coefficients. The code is, therefore, as follows:

```
      DO 2 J = 1, N
        DO 1 K = 1, M
          C( , J) = C( , J) + A( , K) x B(K, J)
1       CONTINUE
2     CONTINUE
```

where C(, J) and A(, K) indicate the columns of C and A respectively. In the above code there is a parallel addition of N elements and the multiplication of scalar B(K, J) by the vector A(, K). Therefore the parallelism exploited is N (the elements of each column of C), compared with 1 for the original conventional code. Furthermore, this type of operations (vector + vector * scalar) are very suitable to be executed on the vector computer ("chaining" of the operations) and it is possible to reach high value of performance.

For computing the solution of OPF problems whose model is the nonlinear program given by (15), a vector supercomputer CRAY X-MP/48 has been used.

The CRAY X-MP/48 is a multiprocessor machine (four processors), in which each processor has a pipeline architecture (same organization as CRAY 1); the four central units share a large core memory unit with interleaved organization (32 banks) and I/O subsystem; the clock cycle is about 8.5 nsec.

In addition to vector capabilities, the multiprocessor organization of CRAY X-MP allows to use efficient multitasking facilities at different programming levels (parallelism among independent subroutines or do loops).

7. NUMERICAL RESULTS

The OPF problem (15) has been taken into consideration for numerical computation in the case of the following networks:

NET 1	N = 5	N_L = 6	N_g = 2	N'_g = 2
NET 2	N = 10	N_L = 13	N_g = 7	N'_g = 7
NET 3	N = 57	N_L = 78	N_g = 4	N'_g = 7

The VMM, in its sequential implementation, has been firstly used on the serial computer IBM 4341; the relevant numerical results are reported in Table 1. On the other hand, in Table 2 the results obtained by using a parallel implementation on the vector supercomputer CRAY X-MP/48 are displayed.

TABLE 1

NETWORK	N	N_V	N_C	I_T	F*	CPU sec
NET 1	5	13	56	5	$.138 \times 10^3$	5
NET 2	10	33	132	19	$.280 \times 10$	91
NET 3	57	124	600	5 low acc.	$.339 \times 10$	38 x 60

TABLE 2

NETWORK	N	N_V	N_C	I_T	F*	CPU sec
NET 1	5	13	56	5	$.138 \times 10^3$	0.122
NET 2	10	33	132	19	.280x10	1.60
NET 3	57	124	600	8 low acc.	.379x10	34.8

In the above tables the following notation is used:

N = number of nodes

N_V = number of variables;

N_C = number of constraints;

I_T = number of iterations;

F* = function value at the computed minimizer.

For the sake of comparison it is worth noting that a remarkable speed-up factor was gained by the vectorized algorithm versus the standard implementation, in terms of CPU time.

The improvement attainable with the vectorized algorithm may be particularly valuable when the network's dimension increases. Thus, a special attention must be devoted to the way in which the code is designed and matched to the architecture of the machine; in fact, only under these conditions, its capabilities can be conveniently and effectively exploited by the complex data-structure typical of large scale problems.

It can be drawn the final consideration that the optimal operation of a complex electric network, even rather large, is successfully computed by a vector machine, thus making possible its "real time" management . The use of a sound nonlinear optimization algorithm is a crucial contribution to pursue this objective in a reliable way.

8. CONCLUDING REMARKS

Looking at the numerical results previously reported, the conclusion can be drawn that the performance of the parallelized VVM algorithm is sharply superior to the sequential one. In particular, the speed-up factor indicates a "jump of quality" in favour of the vectorized

implementation.

This aspect encourages to face bigger problems, whose dimensionality is usually prohibitive for sequential machine implementations.

However, the occasion of this initial attempt to solve OPF problems by using new computational techniques, allows some considerations useful for future works.

The network model used, although quite general does not take into account further real aspects which would make more complex the model.

At present the VVM may suffer for inconsistency of constraints in the quadratic programming subproblem, at least in some critical situations.

Furthemore, it is not taken into account the sparsity structure of the matrices involved in the problem.

Finally, parallel techniques have been applied at level of code design, without changing the structure of the algorithm.

In perspective the potentiality of parallel computation may be fully exploited by designing optimization methods inherently conceived for implementation on multiprocessing computer systems.

This may result in an extraordinary improvement of performances opening the possibility to solve even extremely large scale problems in almost real time.

REFERENCES.

[1] A. Miele, E. M. Sims, V. K. Basapur, J. C. Heideman, *Sequential Gradient Restoration Algorithm for Mathematical Programming Problems with Inequality Constraints.* International Meeting on Optimization Problems in Engineering. Cassino, Italy, 1984.

[2] L. Grandinetti and D. Conforti, *Optimization of Power Flow in Electric Power Systems via Parallel Processing.* Large Scale Systems Theory and Applications, ed. H. P. Geering and M. Mansour, Pergamon Press (London), 840-844, 1987.

[3] M. Avriel and D. J. Wilde, *Optimal Search for a Maximum with Sequences of Simultaneous Function Evaluations.* Management Science 12, 722-731, 1966.

[4] R. M. Karp and W. L. Miranker, *Parallel Minimax Search for a Maximum.* Journal of Combinatorial Theory 4, 19-35, 1968.

[5] D. Chazan and W. L. Miranker, *A Non-gradient and Parallel Algorithm for Unconstrained Minimization.* SIAM Journal on Control 8, 207-217, 1970.

[6] D. A. Pierre, *A Nongradient Minimization Algorithm having Parallel Structure, with Implementation for an Array Processor.* Computational and Electrical Engineering 1, 3-21, 1973.

[7] T. A. Straeter, *A Parallel Variable Metric Optimization Algorithm.* NASA Technical Note D-7329, Langley Research Center, Hampton, Virginia 23665, 1973.

[8] P. van Laarhoven, *Parallel Variable Metric Algorithms for Unconstrained Optimization.* Mathematical Programming 33, 68-81, 1985.

[9] T. A. Straeter and A. T. Markos, *A Parallel Jacobson-Oksman Optimization Algorithm.* NASA Technical Note D-8020, Langley Research Center, Hampton, Virginia 23665, 1975.

[10] J. S. Kowalik and K. G. Ramakrishnan, *A Numerically Stable Optimization Method based on a Homogeneous Function.* Mathematical Programming 11, 50-66, 1976.

[11] D. H. Jacobson and W. Oksman, *An Algorithm that Minimizes Homogeneous Functions of N Variables in N+2 Iterations and Rapidly Minimizes General Functions.* Journal of Mathematical Analysis and Applications 38, 535-552, 1972.

[12] L. C. W. Dixon, *On Nazareth's Three Term Conjugate Gradient Method.* NOC Technical Report 133, Hatfield, Herts., 1983.

[13] L. C. W. Dixon, P. G. Ducksbury and P. Singh, *A New Three Term Conjugate Gradient Method.* NOC Technical Report 130, Hatfield, Herts., 1983.

[14] L. C. W. Dixon, P. G. Ducksbury and P. Singh, *A Parallel Version of the Conjugate Gradient Algorithm for Finite Element Problems.* NOC Technical Report 132, Hatfield, Herts., 1982.

[15] L. C. W. Dixon and K. D. Patel, *The Place of Parallel Computing in Numerical Optimization, Four Parallel Algorithms for Nonlinear Optimization.* NOC Technical Report 125, Hatfield, Herts., 1982.

[16] M. Dayde, *Parallelisation d'Algorithmes d'Optimisation pour des Problèmes d'Optimum Design.* Thèse, Institut National Polytechnique de Toulouse, France, 1986.

[17] B. M. E. de Silva , K. M. Ragsdell, E. Sandgren, V. Bhatt, D. Marjadi, S. J. Wu, *Design of Ultra-Light Vehicles using Decomposition in a Parallel Computing Environment.* Report 86DPC021, Design Productivity Center, College of Engineering, University of Missouri, Columbia, Missouri 65211, 1986.

[18] R. B. Schnabel, *Parallel Computing in Optimization.* In K. Schittkowski (ed.), Computational Mathematical Programming. Springer, Berlin, 357-382, 1985.

[19] F. Lootsma and K. M. Ragsdell, *State of the Art in Parallel Nonlinear Optimization.* Faculty of Mathematics and Informatics, Delft University of Technology, Report N. 86-44, 1986.

[20] S.P. Han, *Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems.* Mathematical Programming 11 , 263-282, 1976.

[21] M. J. D. Powell, *A fast Algorithm for Nonlinearly Constrained Optimization Calculations* . Numerical Analysis, Dundee 1977, Lecture Notes in Mathematics 630, ed. G.A. Watson, Spinger-Verlag (Berlin), 144-157, 1978.

[22] K. Schittkowski, *On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function,* Math. Operationsforschung und Statistik, Ser. Optimization 14 , 197-216, 1983.

[23] M. J. D. Powell, *Variable Metric Methods for Constrained Optimization,* Mathematical Programming-The state of Art, eds. A. Bachem, M. Grotschel and B. Korte, Springer-Verlag (Berlin), 287-311, 1983.

AUTOMATIC DIFFERENTIATION AND PARALLEL PROCESSING IN OPTIMISATION

by

L C W Dixon
Associate Dean (Research)
School of Information Sciences
The Hatfield Polytechnic

Abstract

This paper introduces the concepts of automatic differentiation and parallel processing and discusses how the advent of the ADA language on parallel processing machines may transform the solution of practical optimisation problems.

1. Introduction

The solution of large industrial optimisation problems should produce considerable economic and technological benefits. The mathematical theory necessary to write iterative algorithms for the solution of both unconstrained and constrained optimisation algorithms is well known and codes exist that regularly solve small to medium industrial optimisation problems. These codes are normally successful when applied to problems containing between 5 and 100 variables, but are normally not written with the intent to solve problems containing thousands of variables. Though occasionally they have been successfully used to solve such problems. There is really a surprisingly small amount of optimisation software available which efficiently handles the sparse matrices that nearly always occur in large problems. There is also apparently little industrial demand for such problems.

If we seek for the reason for this lack of demand, I believe it exists not because of a lack of problems but because of a lack of confidence in the ability to solve such an optimisation problem. This lack of confidence partly arises from the difficulties people have experienced in solving more modest problems in the past. Assuming codes are implemented properly, these difficulties have tended to arise in three areas. Most optimisation algorithms that can theoretically be shown to converge need to be given not only accurate algorithms to evaluate the objective function and the constraints but also accurate algorithms and hence expressions to compute their derivatives. This can be difficult for simple test functions in a few variables but can take man months of effort on industrial problems.

Once these derivatives are available two further difficulties have caused problems in the past, namely, the need for sufficient store on the computer to store

second derivative information; and the tendency for solution codes for industrial optimisation to require considerable computer time.

This paper is written with the assumptions that computer store is no longer such a vital problem and that with the decrease in relative costs of computer time to human time, we can afford to increase the solution time on the computer to do away with the man months of effort often needed to obtain accurate derivatives. This is in any case reduced as the availability of accurate derivative information frequently reduces the number of iterations required by optimisation algorithms.

The first topic to be treated in this paper will therefore be automatic differentiation. This technique introduced by Rall [1] enables accurate derivates to be obtained by the computer.

Whilst simple implementations can considerably increase the computer time, the use of the principle of partially seperable subfunctions, Toint [2], can reduce this considerably. In this aproach we decompose

$$F(x) = \sum_i f_i(x_i)$$

where each subfunction f_i only depends on a small number of variables x_i.

One of the most natural decompositions occurs in finite element optimisation Dixon & Ducksbury [3]

$$F(x) = \sum_{el} F_{el}(x_{el})$$

where the structure for automatic differentiation is identical to that used successfully on the ICL-DAP parallel array processor, Ducksbury [4].

2. Automatic differentiation

2.1 Disadvantages of analytic or numeric approximation method

Most algorithms for solving unconstrained problems

$$\text{Min } F(x) \qquad x \in R^n \qquad \text{P1}$$

or constrained problems

$$\text{Min } F(x) \qquad x \in R^n$$

$$\text{s.t. } e_i(x) = 0 \qquad i = 1 \ldots M_E \qquad \text{P2}$$

$$h_j(x) \geq 0 \qquad J = 1 \ldots M_I$$

theoretically require the algorithm to methods for evaluating first derivatives accurately i.e.

$$g_k = \frac{\partial F}{\partial x_k}$$

and $\frac{\partial e_i}{\partial x_k}$ and $\frac{\partial h_j}{\partial x_k}$ when appropriate.

As the same arguments apply to both P1 and P2 we will only discuss the simpler, unconstrained problem P1 in this paper.

For this problem it is well known that the availability of second derivatives

$$G_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j}$$

can produce more efficient iterative methods.

In even 20 dimensions to obtain g accurately implies analytically obtaining 20 difficult formulae and then programming each accurately. To obtain the second derivatives 190 expressions would need to be programmed. Each program would need to be verified.

It has therefore become customary to obtain numeric approximations to the derivatives, by using, say, central difference formula

$$g_k^A = [F(x + ha_i) - F(x - ha_i)]/2h.$$

These formula are of course subject to both truncation errors ($O(h^2)$) and rounding errors ($O(1/h)$). It can therefore be difficult to choose an appropriate value of h. If we are asking g_k^A to verify an analytic expression g_k and the two disagree, it is frequently difficult to decide whether this is due to an error in calculating g_k or the selection of an inappropriate value of h.

The use of automatic differentiation involves less arithmetic than the calculation of g_k^A and avoids these difficulties.

2.2 Rall's method of automatic differentiation

The method of automatic differentiation introduced by Rall calculates the function value, gradient vector and Hessian matrix simultaneously.

Each entity y occurring in a program is stored as a triplet:-

$$\bar{y}_i = (y_i, \underline{y}_i', y_i'')$$

where y_i' is an n vector $\frac{\partial y_i}{\partial x_k}$ $\quad k = 1 \ldots n$

and y_i'' is an n x n vector $\frac{\partial^2 y}{\partial x_k \partial x_l}$ $\quad k = 1 \ldots n$, $\quad l = 1 \ldots n$

i.e. a list of $n^2 + n + 1$ elements.

An algebra of triplets can now be defined which mimic the rules of differentiation

1) $\bar{u} + \bar{v} = (u, \underline{u}', u'') + (v, \underline{v}', v'')$
$= (u + v, \underline{u}' + \underline{v}', u'' + v'')$

2) $\bar{u} - \bar{v} = (u - v, \underline{u}' - \underline{v}', u'' - v'')$

3) $\bar{u}.\bar{v} = (u.v, u\underline{v}' + v\underline{u}', uv'' + uv'^T + vu'^T = vu'')$

4) $\bar{u}/\bar{v} = (u/v, (v\underline{u}' - u\underline{v}')/v^2, (v^2u'' - v(\underline{v}'\underline{u}'^T + \underline{u}'\underline{v}'^T)$
$+ 2u\underline{v}'\underline{v}'^T - uvv'')/v^3$

$v \neq 0.$

Similar rules can be defined by all the standard operations

exp(x), sin(x), cos(x), $\log_e(x)$ etc.

These rules imply that the natural variables are held as $\bar{x}_i = (x_i, \underline{a}_i, \emptyset)$ and constants as $c = (c, \underline{0}, \emptyset)$.

In Fortran the implementation of a function evaluation in triplets is cumbersome as each addition, subtraction, multiplication and division must be programmed by a separate subroutine call. At this point the use of concepts of types and extended operations in the ADA language greatly simplifies the programming. These can be not only used for well known constructs such as matrices, where the definition of a type matrix and an extended operator * can allow the product

C = A*B

to be programmed in one line, but also for new constructs such as Rall Triplets. Mohseninia [5] defines the type triplex and has written a full set of operator extensions to be applied to them.

If X1, X2 and F are triplets then in ADA the statement

F = 100.0*((x$_2$ - X1**2)**2) + (1.0 - X1)**2

given X1 = $(-1.2;\ (1, 0)^T;\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

X2 = $(1.0;\ (0, 1)^T;\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

calculates

F = $(24.2;\ (-215.6, -88.0)^T;\ \begin{pmatrix} 1330 & 480 \\ 480 & 200 \end{pmatrix})$

i.e. the function value, gradient and Hessian of Rosenbrock's function at the classic starting point.

2.3 Automatic differentiation and optimisation

Although we have described automatic differentiation in the previous section as a triplet, containing function, gradient and Hessian; it is obviously trivial to reduce it to a doublet consisting of function and gradient alone.

It would obviously be possible to run any gradient method such as the variable metric method or conjugate gradient method for unconstrained optimisation or the recursive quadratic programming method for constrained optimisation by calculating the first derivatives using the doublet form of automatic differentiation.

Similarly by using the triplet form second order methods such as the modified Newton method, or Dembo-Steihaug truncated Newton method [6,7] for unconstrained optimisation or Maany's QFQC [8] method for constrained optimisation can be used without difficulty.

In practice, due to our experience reported in Dixon & Price [9] that the truncated Newton method is efficient over a wide range of variables, we have

incorporated automatic differentiation into that code, Dixon & Price [10]. An ADA implementation is described in [5].

In their implementation of the truncated Newton method Dembo and Steihaug avoid the calculation of the matrix G by noting that this only appears in their algorithm in the form Gp and this can be approximated by

$$Gp = (g(x + hp) - g(x))/h.$$

In Dixon & Price [10] a modification of the automatic differentiation triplet is described that uses the direction p as an input and evaluates a triplet {F, g, Gp} without using a matrix store. The algebra of such triplets is also given in that paper.

It is however well known that sparse Hessians can frequently be computed by difference techniques in far less than n steps and it can therefore be argued that for such problems Dembo and Steihaug's strategy of calculating Gp at every minor iteration would be far more expensive than calculating G initially in a few gradient calculations and then only forming the product at each iteration. The latter seems especially attractive if G is sparse.

It therefore seemed sensible to seek an alternative triplet that would preserve sparsity.

2.4 Sparse automatic differentiation

Any objective function calculation must be made up of a number of subfunction calculations. These subfunctions will normaly each depend simply on a subset of the variables. The Hessians of these subfunctions will therefore usually be sparse. This is especially true if overall the Hessian is sparse.

This idea has already been utilised by Toint [2] in his partially seperable function theory

$$F(x) = \sum_i f_i(x_i)$$

and Dixon & Ducksbury [3] in finite element optimisation

$$F(x) = \sum_{el} F_{el}(x_{el}).$$

Automatic differentiation has therefore been further modified, in that associated with each 'sparse triplet' definition is a list of the indexes of those variables that affect that subfunction. If there are K such variables, the gradient part of the 'sparse triplet' is a K vector and the Hessian part consists of K linked lists. These linked lists which use the Dynamic data structure and are updated using the recursive facilities available in ADA, are described in detail in [5]. In each row of the sparse matrix, the structure states in which column the particular nonzero element exists, its value and the position of the next element in the row.

By computing the Hessian in this sparse way the amount of arithmetic is kept to a minimum during the calculation and the products Gp used in the truncated Newton method can be computed very efficiently.

Using the 'sparse triplet' approach the calculations involved in even small compact functions are reduced.

We therefore now recommend the calculation of the objective function by the 'sparse triplet' method at the start of each outer iteration of the truncated Newton method; and simply forming the product Gp in each inner iteration.

We have not been able to show any relationship between the amount of arithmetic involved in such a sparse triplet calculation and the amount needed to form a sparse Hessian by the Curtis, Powell & Reid [11] method. This question is still open.

Six combinations of automatic differentiation and the truncated Newton method have been investigated. Four in Fortran and two in ADA, namely:-

1) Compute g and G by Rall's triplet method
2) Compute g by Rall's doublet method and Gp by the finite difference approximation
3) Compute g and Gp by the modified triplet, Dixon & Price [10]
4) Compute g and G by 'partial separation' and submatrices, Dixon & Price [12]
5) Compute g and G by 'sparse triplet' (ADA), Mohseninia [5]
6) Compute g and Gp by 'sparse modified triplets', Mohseninia [5].

An automatic differentiation implementation of the Gauss Newton method for minimising a sum of least squares with small residuals is described in Dixon & Price [13].

Toint [14] gives an efficient code for large residual problems. This could be implemented using automatic differentiation very easily.

3. Parallel Processing

Many optimisation problems can be decomposed into subfunctions

$$F = \sum_{el} F_{el}(x_{el})$$

where

$$\nabla F = \sum_{el} \nabla F_{el} \quad \text{where } \nabla F_{el} \text{ is essentially an } n_{el} \times 1 \text{ vector}$$

and

$$\nabla^2 F = \sum_{el} \nabla^2 F_{el} \quad \text{where } \nabla^2 F_{el} \text{ is essentially an } n_{el} \times n_{el} \text{ matrix.}$$

This is of course a similar structure to that used in 'sparse triplet' automatic differentiation. This is the same structure used by Ducksbury [4] in finite element optimisation on the ICL-DAP. He was solving a set of nonlinear partial differential equations and converted this into a nonlinear optimisation problem. Both were then approximated using standard finite element techniques and each element was calculated on its own processor.

One sample problem considered was

$$e(u) = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} - Ru\frac{\partial u}{\partial x_1} = 0$$

s.t. $4 \leq x_1 \leq 5$; $4 \leq x_2 \leq 5$.

The solution of this problem is $u = 6x_1/Rx_2$.

The problem can be converted to an optimisation problem

$$\min_u F(u) = \iint_A e(u)^2 dA.$$

Obviously the global minimum of F(u) = 0 and this occurs at the solution to the partial differential equation. Ducksbury introduced 3 fields A, B, C

$$A = u,\ B = \frac{\partial u}{\partial x_1},\ C = \frac{\partial u}{\partial x_2}$$

and a function

$$F = \iint_A e_1^2 + e_2^2 + e_3^2\ dA$$

where

$$e_1 = \frac{\partial B}{\partial x_1} + \frac{\partial}{\partial x_2} - RAB = 0$$

$$e_2 = B - \frac{\partial A}{\partial x_1} = 0$$

$$e_3 = C - \frac{\partial A}{\partial x_2} = 0.$$

Each field A, B, C was approximated by a standard bilinear element over rectangles, which led to $N_{EL} = 12$ even for fine meshes with over 4800 variables.

It is very intriguing to note that if standard linear elements had been used over triangular elements he would have had $N_{EL} = 9$; and for triangular elements with quadratic approximations over u then $N_{EL} = 6$.

Ducksbury's results on the DAP were obtained using analytic differentiation but the structure used in Automatic Differentiation, method 4, and parallel computation are identical.

We can therefore predict a future where automatic differentiation in ADA will be implementable on parallel processing machines using its parallel tasking facilities.

Conclusions

Automatic Differentiation can be used to obtain derivative calculations for first order methods like variable metric and conjugate gradient methods.

It can be combined very efficiently with the Truncated Newton method.

Efficient methods for computing sparse Hessian matrices of large problems can be easily implemented especially in ADA using the 'sparse triplet' approach.

The method used for sparse matrix calculations of finite element optimisation problems are very similar to those previously used to solve such programs on the parallel processing array (ICL-DAP) machine.

We can anticipate being able to combine automatic differentiation and parallel processing in ADA to solve large problems.

References

1. Rall, L B, Automatic Differentiation: Techniques and Applications, Lecture Notes in Computer Science No. 20, Springer Verlag, 1981.

2. Toint, P, On Large Scale Nonlinear Least Squares Calculations, University of Namur, Belgium, 1985.

3. Dixon, L C W, Ducksbury, P G and Patel, K D, Experience Running Optimisation Algorithms on Parallel Processing Systems, presented as plenary session at 11th IFIP Conference on System Modelling and Optimisation, Copenhagen, 1983.

4. Ducksbury, P G, Parallel Array Processing, Ellis-Horwood, 1986.

5. Mohseninia, M, The Use of Extended Operations Set of ADA with Automatic Differentiation and the Truncated Newton Method, Technical Report No. 176, Numerical Optimisation Centre, The Hatfield Polytechnic, 1987.

6. Dembo, R, Eisenstat, S C and Steihaug, T, Inexact Newton Method, SIAM Journal of Numerical Analysis, Vol. 19, pp 400-408, 1982.

7. Dembo, R, and Steihaug, T, Truncated Newton Methods for Large Scale Optimisation, Mathematical Programming, Vol. 26, pp 190-212, 1983.

8. Maany, Z A, A New Algorithm for Highly Curved Constrained Optimisation, Mathematical Programming Study on Computational Mathematical Programming, North Holland (to appear).

9. Dixon, L C W and Price, R C, Numerical Experience with the Truncated Newton Method, JOTA (to appear).

10. Dixon, L C W and Price, R C, The Truncated Newton Method for Sparse Unconstrained Optimisation Using Automatic Differentiation, JOTA (to appear).

11. Curtis, A R, Powell, M J D and Reid, J K, On the Estimation of Sparse Jacobian Matrices, J. Inst. Math. Appl., Vol. 13, pp 117-119, 1974.

12. Dixon, L C W, Dolan, P and Price, R C, Finite Element Optimisation: The Use of Structured Automatic Differentiation, Proceedings of IMA Conference on simulation and Optimisation of Large Systems, University of Reading, 1986.

13. Dixon, L C W and Price, R C, The Truncated Gauss-Newton Method for Sparse Least Squares Problems Using Automatic Differentiation, Technical Report No. 171, Numerical Optimisation Centre, The Hatfield Polytechnic, 1986.

14. Toint, P, An Introduction to Quasi-Newton Methods for Large Scale Unconstrained Nonlinear Programming, University of Namur, Belgium, 1986.

MULTICRITERIA OPTIMIZATION IN THE DISO SYSTEM

Evtushenko Y., Mazourik V., Ratkin V.
Computing Center of the USSR Academy of Sciences
Moscow, USSR

Preface. From the mathematical point of view multicriteria optimization (MCO) is a natural generalization of optimization problems. The need of decision making in contradictory situations makes MCO methods so interesting for us. MCO deals with one of the most sophisticated aspects of human activity which is to achieve several goals by the single act of decision making. MCO models and ordinary optimization are not very much different in task definition, giving us hope to use the similar numerical methods.

This paper gives the overview of the MCO package as one of the main parts of the DISO - dialogue system for optimization problem solving which was developped in the Computing Center of the USSR Academy of Sciences. Two MCO methods are described in this paper. Both methods are based on the idea of nonuniform covering technique and inclusion function approach, which was initially developped for global extremum search [1-5]. The complexity of MCO tasks makes it necessary to create effective numerical methods to find both a single point of Pareto set and an approximation of this set also. The paper describes two MCO algorithms which differ in the interpretation of the solution and as a consequence in the complexity of numerical calculations. The main features of the MCO package are described also.

1. Overview of the DISO system.

The basic feature of the DISO system is the integration principle. Unlike other dialogue systems of this class the DISO system includes several interconnected packages for the solution of the following tasks:

- unconstrained minimization;
- nonlinear programming;
- optimal control;
- linear programming;
- global optimization;

- multicriteria optimization;
- linear algebra;
- nonlinear algebraic equations.

For all these tasks the DISO system delivers for the user the unified set of dialogue capabilities, which includes:

- task definition and analysis in text mode;
- automatic optimization class recognition and correspondent dialogue package initiation;
- local analysis of task definition functions in a given point;
- changing of optimization method and its control parameters in the dialogue session;
- asynchronous control of the solution process which makes it possible to stop the process at any moment;
- control of the hierarchical solution process with the automatic or manual creation of the subordinate tasks from the list mentioned above;
- choice of the numerical or analytical differentiation schemes;
- control of the numerical and graphical interaction in the process of solution;

From the implementation point of view dialogue capabilities of the DISO system are based on the new approach of the multiwindow technique which includes the capability to access the values of the variables using special fields in the windows. The methods in each optimization class have different forms of graphical output in accordance with their basic mathematical schemes.

The integrated mode of the DISO system makes it really powerfull instrument for the solution of different application tasks. It is essential, that the system makes it possible to change the task definition in the process of the solution and to transfer the task from one mathematical model to another. The typical example of such transformation takes place if you add a restrictions to the initial unconstrained optimization problem thus creating nonlinear programming problem. Another example: transforming all the criteria but one to the restrictions will change the MCO problem to the more simple class of nonlinear programming. It is important to note that all the numerical results achieved so far will remain accessible after these transformations.

Most optimization models have hierarchical structure. The nonlinear programming methods, for example, may reduce the task to unconstrained

minimization on every iteration. Optimal control methods usually create subordinate tasks of nonlinear programming and so on. In all these cases the DISO system provides the dialogue capabilities of the correspondent class for the solution of the subordinate problem. Finishing the solution will move the user again to the level of the initial problem. This feature of the DISO system proved to be really valuable for the solution of several difficult applied problems saving time and increasing the accuracy of the solution.

The MCO dialogue package plays a central role in the DISO system, because its mathematical model can be obviously treated as a generalization of the other optimization models listed above.

2. Statement of MCO problem.

The multicriteria problem that we consider has the following form:

$$\min_{x \in X} F(x), \qquad (1)$$

where $x \in R^n$ is a vector of decision variables, X is the feasible decision set, $F=[F^1, \ldots, F^m]$ is the objective vector, $F: R^n \to R^m$, vector-function F is continuous on X. The decision set X is assumed to be a closed and bounded (therefore compact). The goal is to find the efficient (Pareto) set X_* of X with respect to F, that is

$$X_* = \{x \in X: \text{ if } F(w) \leqslant F(x) \text{ for some } w \in X \text{ then } F(w)=F(x)\}. \qquad (2)$$

We shall use the following convention: if $a,b \in R^s$ then $a \leqslant b$ if and only if $a^i \leqslant b^i$ for all $i=\overline{1,s}$.

We propose two extentions of ε-optimality concept which were developed for scalar optimization problem to vector case. In the first extention we introduce ε-efficient set as follows:

$$X_*^\varepsilon = \{x \in X: F(x) \leqslant F(x_*)+\varepsilon, \text{ where } x_* \in X_*\}, \qquad (3)$$

where vector $\varepsilon \in R^m$ has all positive components and is named accurasy vector.

The set W_*^ε is called ε-net of the Pareto set iff:

1) for any point $x \in X_*$ there exists a point $z \in W_*^\varepsilon$ such that $F(z) \leqslant F(x)+\varepsilon$;
2) there are no two different points x and z in W_*^ε such that $F(x) \leqslant F(z)$.

Let X and P_i be a compact right parallelepipeds parallel to the coordinate axis (abbreviated as a box in the sequel):

$$X=\{x \in R^n: a \leqslant x \leqslant b\},$$

$$P_i=\{x \in R^n: a_i \leqslant x \leqslant b_i\}, \; P_i \subset X, \; a_i \in R^n, \; b_i \in R^n.$$

The main diagonal of the box P_i we denote as $d_i=b_i-a_i$, the midpoint of the box is $c_i=1/2(b_i+a_i)$.

Let's introduce the m-dimentional vector-function Q(P), for which every j-th component is defined by the condition:

$$Q^j(P) = \min_{x \in P} F^j(x), \quad P \subset X.$$

We assume that for Q(P) it is possible to find vector-function G(P) which is the lower estimation of Q(P) on the box P. This function must satisfy two conditions:

$$G(P) \leq Q(P), \tag{4}$$

$$\lim_{\|d_i\|_\infty \to 0} (G(P_i)-Q(P_i))=0. \tag{5}$$

Here we introduced the sequence of the boxes which satisfies the following conditions:

$$P_{i+1} \subset P_i, \quad \lim_{i \to \infty} P_i = P_\infty \in X,$$

where P_∞ is accumulating point.

For the given vector-function F(x) the vector-function G(P) can be found either on the basis of interval analysis [6] or by introducing some additional hypothesis. For example, supposing that all the components of F(x) on the X set satisfy the Lipschitz condition with constants $L^j = \max_{1 \leq i \leq n} \max_{x \in X} |F^j_{x^i}(x)|$ than we have:

$$G^j(P_i)=F^j(c_i)-1/2L^j \|d_i\|_\infty. \tag{6}$$

If in addition F(x) satisfies Lipschitz condition with constant M then:

$$G^j(P_i)=F^j(c_i)-1/2\|d_i\|_\infty \min\{L^j, \|F'_x(c_i)\|_1+1/4M\|d_i\|_\infty\}. \tag{7}$$

It is obvious that these functions G(P) satisfy conditions (4), (5).

3. Description of MCO methods.

Now we are going to describe two algorythms for the approximate solution of the problem (1). During the computation process the algorythms will generate the sequence of the boxes

$$B_k=\{P_1, P_2, \ldots, P_k\}, \text{ all } P_i \subset X.$$

Let each box P_i be linked with the structure $S_i=(c_i, d_i, G_i)$, where $G_i=G(P_i)$. We call the set S for the sequence B_k the structure list $S=\{S_1, S_2, \ldots, S_k\}$.

Algorythms differ in the interpretation of the solution and time consuming. W^ε_* set is obtained by one of them (second algorythm) and a single point from the X^ε_* set - by another (first algorythm). The general scheme of both algorythms is described below.

Algorithm.

Initial actions:

1). Let $P_1=X$, calculate c_1, d_1, $F(c_1)$, $G_1=G(P_1)$ and set $B_1=\{P_1\}$, $S_1=(c_1, d_1, G_1)$, $S=\{S_1\}$.

Main cycle:

2). Choose the box P_s from B_k, for which $\min\limits_{1\le i\le k} \max\limits_{1\le j\le m} G_i^j$ is achieved.

3). Choose a coordinate direction t in box P_s, parallel to which P_s has an edge of maximum length, i.e. $d_s^t = \max\limits_{1\le i\le n} d_s^i$. Bisect P_s in the direction t, getting boxes P_α, P_β with midpoints c_α, c_β and diagonals d_α, d_β respectively.

4). Calculate $F(c_\alpha)$, $F(c_\beta)$, modify W_*^ε or X_*^ε and define the vector values $G_\alpha=G(P_\alpha)$, $G_\beta=G(P_\beta)$.

5). Remove the box P_s from the sequense B_k, i.e. remove the structure S_s from the list S. Add into the list S two structures: $S_\alpha=(c_\alpha, d_\alpha, G_\alpha)$ and $S_\beta=(c_\beta, d_\beta, G_\beta)$, assuming $S_s=S_\alpha$ and $S_{k+1}=S_\beta$.

6). For all S_i from the list S check the following condition: if $F(x_j)\le G(P_i)+\varepsilon$ for any x_j from W_*^ε or X_*^ε then remove S_i from the list S and remove P_i from B_k. Order new list $\{S_{i_1}, S_{i_2}, \ldots, S_{i_p}\}$ and give it the name $S=\{S_i\}_{1\le i\le p}$.

7). Let $k=p$. If $k\neq 0$, i.e. S is not empty then goto 2.

Concluding operations:

8). Invoke output procedure. Stop computation.

4. Problem solving in MCO package.

The task below was solved in order to investigate the quality of MCO algorithms. Full definition of this task is given in [7]. The solution based on the sequence of minimization subtasks was found there. This task includes two criteria, five parameters and has the following form:

$$\min F(x), \quad x \in X$$

where $F^1(x)=1-\prod_{i=1}^{n}\left(1-\left(1-r^i\right)^{x^i+1}\right)$, $F^2(x)=\sum_{i=1}^{n} c^i x^i$,

$X=\{x^i \in N: 0\le x^i\le 10,\ i=\overline{1,n}\}$, $n=5$, vectors c and r are given in the Table 1.

Two solutions were obtained using both algorithms with different accuracy vectors ε. G vector function was defined in accordance to (6). Lipshits constants vector was chosen to be $L=[0.7\ 0.7]$. This value is

good higher estimation for the given vector function. The results for the first algorithm are given in Table 2 for different accuracies. Table 3 contains the results for the second algorithm with the accuracy vector $\hat{\varepsilon}=[0.1\ 0.35]$.

Table 1

i	1	2	3	4	5
c	0.13	0.13	0.15	0.14	0.15
r	0.90	0.75	0.65	0.80	0.85

Table 2

accurasy		criteria		parameters					crit.
ε^1	ε^2	$F^1(x)$	$F^2(x)$	x^1	x^2	x^3	x^4	x^5	evaluations
0.20	0.50	0.575	0.43	0	0	0	2	1	163
0.15	0.40	0.444	0.44	0	0	1	1	1	335
0.10	0.35	0.336	0.55	1	1	1	1	0	591

Table 3

N	criteria		parameters				
	$F^1(x)$	$F^2(x)$	x^1	x^2	x^3	x^4	x^5
1	0.096	1.12	1	2	2	2	1
2	0.125	0.98	1	2	2	1	1
3	0.166	0.85	1	1	2	1	1
4	0.198	0.83	1	2	1	1	1
5	0.236	0.70	1	1	1	1	1
6	0.305	0.57	0	1	1	1	1
7	0.336	0.55	1	1	1	1	0
8	0.396	0.42	0	1	1	1	0
9	0.446	0.41	1	1	1	0	0
10	0.497	0.28	0	1	1	0	0

5. MCO package dialogue capabilities.

As was already mentioned above, the DISO system provides the user the unified set of dialogue capabilities to control every step of the solution process from the task definition up to the analysis of the numerical results.Let's overview briefly these capabilities.

The MCO task definition can be done using text processor which is part of the DISO system. A special language DIFALG is used for task definition. This language is very similar to the ALGOL-60. The inner form of the function representation is created as a result of the task definition compilation process. The calculation of the numerical values of the functions in a given point is based then on the interpretation of this inner form. The peculiarity of the DIFALG language is that its semantics includes the notion of the differentiation. Function evaluation may be done in parallel with the first and second derivatives of this function in a given point by the user request. The important point is that it gives the user not the numerical approximation but the exact value of the derivatives, which corresponds to the analytically evaluated value. The differentiation algorithms are based on a special highly effective approach which qualitatively increases the calculation speed.

Task definition includes comment lines. The system automatically defines the optimization class and passes the control to the correspondent dialogue monitor analysing these comments. For example, to define MCO problem, the user marks with the special comment those functions in the DIFALG listing which he wants to be included to the set of the criteria. The other comments mark the functions to be included to the equality and nonequality restriction sets. If any, the problem will be classified as nonlinear (single or multicriteria) programming model. Special comments define the mode of optimization: local or global, give initial point value, parallelepipedal restrictions in the parameter space etc. The user may return to the task definition step at any moment of the solution, make some modifications and continue the solution process.

The user initiates the dialogue session finishing the task definition. The set of control windows become available to him at this moment. Each window provides the user with some resources to control the solution process. The MCO dialogue monitor opens the access to the three windows.

The first one provides the capabilities for manual analysis of the functions local properties in a given point. It also controls the MCO method choice, initiation and termination. As was mentioned above, the DISO windows are structured in a sence that they include the different sets of fields to access the variables and control their values. The first MCO window contains the fields for the decision vector, criteria values vector, list of methods available etc. The user may move the cursor to the parameter vector and give this vector some initial value. The cursor movement to the criteria field automatically yields in the recalculation of the criteria functions in a given point with immediate output of the numerical results in this field. The field which is connected with the list of available methods plays the role of the menu. Each method has its own list of control parameters. The choice of the method automatically provides the user with the new list of fields for control parameters which appear in the window and become accessible by the ordinary routine of the cursor movement. Another field is responsible for the type of the solution: the user can choose ε-approximation of the Pareto set or single-point solution as it was mentioned above. Finally, the window has the menu field to run the method, stop it and switch to the task modification mode.

The second window provides the user with the view of the Pareto set in the process of its creation. With the help of the fields in this window the user can point out two coordinate axis to define the two-dimentional plain to create the projection of the pareto set. The solution process will show each new Pareto point in the special graphical field of this window. Finishing the solution process the user may enter the mode of Pareto set analysis. The cursor will take the form of the arrow in the Pareto field. The user can move this arrow from one Pareto point to another, vizualising correspondent numerical values of criteria and parameter vectors. The user can store the Pareto set in a file with a given name or retrieve previously defined Pareto set and continue the solution to achieve more accurate Pareto approximation.

The third window gives the possibility to vizualise the covering technique in the parameter space. Each method has its own covering strategy. The user can vizually estimate the efficiency of the chosen covering scheme for the given task to be solved.

The system provides the possibility of asynchronous control which includes, for example, switching from one window to another while the optimization process continues it's progress.

Conclusion. The MCO methods described in this paper were tested using several tasks and proved to be rather effective. There exist several reasons of the methods success. First, unlike the known methods, these methods are strongly oriented on the effective use of the computer memory which decreases the amount of function evaluation. This is most valuable feature if function evaluation takes long time. Second, the interval analysis technique makes it possible to eliminate the initial Lipshitz constant estimation. Third, there exists a simple and natural way to organize parallel calculations by the feasible domain division between several processors. Fourth, the described methods permit the inclusion of local search algorithms, which may speed the calculations enormously. Fifth, the second of the proposed methods creates the Pareto set using nonunified one-way covering technique, instead of the common approach, which is based on the manifold solution of the auxiliary tasks on the feasible domain.

Dialogue MCO package, which is part of the DISO system, is a promising numerical basis for the implementation of numerous decision support systems.

References.

1. Evtushenko, Y., 1971. Numerical method for global search. Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki, 11 (6): 1390-1403.
2. Evtushenko, Y., 1985. Numerical optimization technique. Schpringer Verlag, New-York. 3. Evtushenko, Y., Potapov, M., 1984. A nondifferentiable approach to multicriteria optimization. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 255: 97-102.
4. Ratschek, H., 1985. Inclusion functions and global optimization. Mathematical Programming. 33 (3): 300-317.
5. Evtushenko, Y., Ratkin, V., 1987. Bisection method for global optimization (in Russian). Izvestija Akademii Nayk SSSR, Tehnicheskaya Kibernetika (to be published).
6. Moore, R.E., 1966. Interval analysis. Prentice-Hall, Englewood Clifs.
7. Saaty T.L. Optimization in integers and related extremal problems. 1970, McGraw-Hill, New York.

Serial and Parallel Solution of Large Scale Linear Programs by Augmented Lagrangian Successive Overrelaxation[1)]

R. De Leone & O. L. Mangasarian

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Technical Report #701

June 1987

Abstract. Serial and parallel successive overrelaxation (SOR) methods are proposed for the solution of the augmented Lagrangian formulation of the dual of a linear program. With the proposed serial version of the method we have solved linear programs with as many as 125,000 constraints and 500,000 variables in less than 72 hours on a MicroVax II. A parallel implementation of the method was carried out on a Sequent Balance 21000 multiprocessor with speedup efficiency of over 65% for problem sizes of up to 10,000 constraints, 40,000 variables and 1,400,000 nonzero matrix elements.

Key Words: Linear programming, SOR, augmented Lagrangian, parallel algorithms

Abbreviated Title: SOR Solution of Linear Programs

[1)] This material is based on research supported by National Science Foundation Grants DCR-8420963 and DCR-8521228 and Air Force Office of Scientific Research Grants AFOSR-86-0172 and AFOSR-86-0255.

1. Introduction

In [8, 9, 10, 12,] successive overrelation methods are proposed for solving the dual of the problem of finding the least 2-norm solution of a linear program. This leads to an exterior penalty formulation of the dual of the original linear program with the interesting property that the penalty parameter need not approach zero in order to obtain an exact solution of the primal linear program [1, 14]. Thus the penalty parameter need only be less than a certain threshold value in order to obtain an exact solution to the primal linear program. However, the penalty parameter must approach zero in order to obtain a solution to the dual problem. Although this approach has been used effectively in conjunction with successive overrelaxation methods both on serial [10] and parallel machines [11, 13], we propose here the use of an augmented Lagrangian on the dual problem instead of an exterior penalty function in order to alleviate the twin difficulties of determining the threshold value of the penalty parameter required for an exact primal solution, and the need for the penalty parameter to approach zero in order to obtain a dual solution. The first proposal for using an augmented Lagrangian formulation for solving linear programs was made in [22]. In [18] Polyak and Tretiyakov made the remarkable discovery that after a finite number of steps of the augmented Lagrangian algorithm, an exact solution to the primal and dual linear programs is obtained. In [3] Golshtein proposed a projected Gauss-Seidel method in conjunction with an augmented Lagrangian formulation and gave computational results for linear programs with sizes up to 352 variables and 166 constraints. No convergence proofs of the projected Gauss-Seidel method was given in [3], nor of the closely related iterative method of Syrov and Churkreidze in [18]. We propose here the use of a projected successive overrelaxation method in conjunction with an augmented Lagrangian formulation. The convergence of the projected SOR method established in [7] is general enough to cover both a serial and a parallel implementation of the method. Since SOR methods are inherently serial in nature, their parallelization is not a routine matter. In [11, 13] two related methods were proposed for the parallelization of SOR methods. The more recent method [13] utilizes an unreduced relaxation factor interval of (0, 2) which we shall employ here with an augmented Lagrangian algorithm for the dual linear program.

The paper is organized as follows. In Section 2 we give the necessary theoretical

background and convergence results for the proposed augmented Lagrangian method applied to the dual linear program. In Section 3 we describe a serial SOR implementation of the method and establish its convergence. In Section 4 we describe our parallel SOR implementation and in Section 5 we present computational results for both the serial and parallel methods.

We briefly describe our notation now. For a vector x in the n-dimensional real space R^n, x_+ will denote the vector in R^n with components $(x_+)_i = \max\{x_i, 0\}$, $i = 1, \ldots, n$. The scalar product of two vectors x and y in R^n will be simply denoted by xy. For $1 \leq p \leq \infty$, the p-norm $\left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$ of a vector in R^n will be denoted by $\|x\|_p$. For the 2-norm the subscript 2 will be dropped. R^n_+ will denote the nonnegative orthant or the set of points in R^n with nonnegative components, while $R^{m \times n}$ will denote the set of all $m \times n$ real matrices. For $A \in R^{m \times n}$, A^T will denote the transpose, A_i will denote the ith row, A_{ij} the element in row i and column j, and for $I \subset \{1, \ldots, m\}$, $J \subset \{1, \ldots, n\}$, A_I will denote the submatrix of A with rows A_i, $i \in I$, while A_{IJ} will denote the submatrix of A with elements A_{ij}, $i \in I$, $j \in J$. Similarly for $x \in R^n$ and $I_\ell \subset \{1, \ldots, n\}$, x_{I_ℓ} will denote x_i, $i \in I_\ell$. The set $\{I_1, I_2, \ldots, I_K\}$ is said to be a consecutive partition of $\{1, \ldots, n\}$ if it is a partition of $\{1, \ldots, n\}$ such that $i < j$ for $i \in I_\ell$, $j \in I_{\ell+1}$ and $\ell = 1, \ldots, k-1$. Here and throughout the symbols $:=$ and $=:$ denote definition of the term on the left and right sides of each symbol respectively.

2. Theoretical Background

We consider the linear program

$$\text{(2.1)} \qquad \min \ cx \quad \text{subject to} \quad Ax \geq b, \ x \geq 0$$

where $c \in R^n$, $b \in R^m$ and $A \in R^{m \times n}$ and its dual

$$\text{(2.2)} \qquad \max_{(u,v) \geq 0} bu \quad \text{subject to} \quad v = -A^T u + c$$

For simplicity we exclude trivial constraints with $A_i = 0$. In [8, 9] the exterior penalty problem associated with the dual problem (2.2)

$$\text{(2.3)} \qquad \max_{(u,v) \geq 0} \varepsilon bu - \frac{1}{2} \left\| A^T u + v - c \right\|^2$$

was solved by an SOR procedure for a sufficiently small value of the penalty parameter ε to obtain $\big(u(\varepsilon), v(\varepsilon)\big)$. The unique least 2-norm solution $\bar{x}$ of the linear program (2.1) was obtained by using the equation

$$\text{(2.4)} \qquad x(\varepsilon) = \frac{1}{\varepsilon} \big(A^T u(\varepsilon) + v(\varepsilon) - c \big)$$

which relates an optimal solution $\big(u(\varepsilon), v(\varepsilon)\big)$ of the dual penalty problem (2.3) and the unique solution $x(\varepsilon)$ of the corresponding quadratic primal problem [5]

$$\text{(2.5)} \qquad \min \ cx + \frac{\varepsilon}{2} xx \quad \text{subject to} \quad Ax \geq b, \ x \geq 0$$

In particular it follows [14] that the least 2-norm solution $\bar{x}$ of the linear program (2.1) is related to $x(\varepsilon)$ of (2.4) by

$$\text{(2.6)} \qquad \bar{x} = x(\varepsilon) \quad \text{for} \quad \varepsilon \in (0, \bar{\varepsilon}] \quad \text{for some} \quad \bar{\varepsilon} > 0$$

Thus the penalty parameter ε of the dual penalty problem (2.3) is the perturbation parameter of the perturbed primal problem (2.5). In order to avoid possible difficulties associated with determining the threshold value $\bar{\varepsilon}$ we consider instead of the exterior penalty problem (2.3) the augmented Lagrangian associated with the dual linear program (2.2)

$$\text{(2.7)} \qquad L(u, v, x, \gamma) := bu - \frac{1}{2\gamma} \left\| A^T u + v - c \right\|^2 - x(A^T u + v - c)$$

It is a standard result [6, 2, 19] that for any $\gamma > 0$, a primal-dual solution $(\hat{x}, \hat{u}, \hat{v})$ of (2.1)-(2.2) is equivalent to a stationary point of the following saddlepoint problem of (2.7): Find an $(\hat{x}, \hat{u}, \hat{v}) \in R^n \times R^m_+ \times R^n_+$ such that for all $x \in R^n$ and all $(u, v) \in R^m_+ \times R^n_+$,

$$L(u, v, \hat{x}, \gamma) \leq L(\hat{u}, \hat{v}, \hat{x}, \gamma) \leq L(\hat{u}, \hat{v}, x, \gamma) \tag{2.8}$$

The standard augmented Lagrangian algorithm [2, 19] consists of a maximization step in the (u, v) space $R^m_+ \times R^n_+$ followed by an unconstrained gradient descent step in the x space R^n. In particular we have the following.

2.1 Augmented Lagrangian Algorithm

Start with any $x^0 \in R^n$. Having x^i determine x^{i+1} as follows

$$\begin{cases} \text{(a) } L(u^i, v^i, x^i, \gamma^i) = \max\limits_{(u,v)\geq 0} L(u, v, x^i, \gamma^i) \\ \text{(b) } x^{i+1} = x^i - \dfrac{1}{\gamma^i} \nabla_x L(u^i, v^i, x^i, \gamma^i) = x^i + \dfrac{1}{\gamma^i}(A^T u^i + v^i - c) \end{cases} \tag{2.9}$$

where $\{\gamma^i\}$ is any bounded sequence of positive numbers.

For this iterative linear programming algorithm Polyak and Tretiyakov [18] have given the following important finite convergence theorem.

2.2 Augmented Lagrangian Algorithm Finite Termination Theorem [18]

For any bounded positive sequence $\{\gamma^i\}$ and $x^0 \in R^n$, Algorithm 2.1 is finite, that is there exists an integer k such that (x^k, u^k, v^k) solve the dual linear programs (2.1)-(2.2). Furthermore for each x^0 there exists a $\hat{\gamma} > 0$ such that for $0 < \gamma^0 < \hat{\gamma}$, the method will terminate in one step, that is (x^1, u^1, v^1) will solve the dual linear programs (2.1)-(2.2).

Note that in the above theorem, two exact maximizations over the (u, v) space $R^m_+ \times R^n_+$ are required in order to obtain $(x^{i+1}, u^{i+1}, v^{i+1})$ from x^i.

We note that since by duality theory [5]

$$\max_{(u,v)\geq 0} L(u, v, x, \gamma) = \min_x \left\{ cz + \frac{\gamma}{2} \|z - x\|^2 \middle| Az \geq b,\ z \geq 0 \right\} =: \phi(x) \tag{2.10}$$

the Augmented Lagrangian Algorithm 2.1 is equivalent to the following gradient method applied to the proximal point function $\phi(x)$

$$\begin{cases} x^{i+1} = x^i - \dfrac{1}{\gamma^i} \nabla \phi(x^i) = \text{Prox}\,(x^i) \\ \text{where Prox}\,(x^i) \text{ is the solution of} \\ \min\limits_z \left\{ cz + \dfrac{\gamma^i}{2} \|z - x^i\|^2 \middle| Az \geq b,\ z \geq 0 \right\} \end{cases} \tag{2.11}$$

Bertsekas [1] and Rockafellar [21] also obtain finite termination for (2.11) from proximal point theory considerations for the linear programming case.

With this background we are prepared now to state and prove the convergence of our serial and parallel SOR algorithms.

3. Serial Successive Overrelaxation Algorithm

The proposed serial algorithm consists of applying the projected SOR method of [7] to the maximization step (2.9a) of the Augmented Lagrangian Algorithm 2.1 for a decreasing sequence of positive numbers $\{\gamma^i\}$. It follows from the Finite Termination Theorem 2.2 and a theorem of Pang [16, Theorem 3.1], that for any x^i the projected SOR method will generate a sequence of points converging to an x^{i+1} that solves the primal linear program (2.1), provided that γ^i is sufficiently small. There are no easily implementable and theoretically justifiable ways of determining how to choose γ^i sufficiently small [20, 4], however we shall prescribe some computationally effective ways for doing that.

The serial projected SOR method has been proposed [7, 9] for solving the quadratic minimization problem

$$\min_{z\geq 0} \theta(z) := \min_{z\geq 0} \frac{1}{2} zMz + qz \tag{3.1}$$

where $M \in R^{k\times k}$ is symmetric and positive semidefinite. This is precisely problem (2.9a) of Algorithm 2.1 if we make the identifications

$$M := \frac{1}{\gamma^i}\begin{bmatrix} A \\ I \end{bmatrix}\begin{bmatrix} A^T & I \end{bmatrix}, \quad q := \begin{bmatrix} A(x^i - \frac{c}{\gamma^i}) - b \\ x^i - \frac{c}{\gamma^i} \end{bmatrix}, \quad z := \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.2}$$

The projected SOR algorithm consists of the following.

$$\begin{cases} z_j^{t+1} = \Big(z_j^t - \omega\big(\nabla^2\theta(z^t)\big)_{jj}^{-1}\nabla_{z_j}\theta\big(z_1^{t+1},\ldots,z_{j-1}^{t+1},\, z_j^t,\ldots,z_k^t\big)\Big)_+ \\ \qquad\qquad\qquad\qquad \omega \in (0,\,2),\ j = 1,\ldots,k \end{cases} \tag{3.3}$$

More specifically for $\theta(z) = \frac{1}{2} zMz + qz$ we have the following.

3.1 Serial SOR Algorithm for $\min_{z\geq 0} \frac{1}{2}\mathbf{zMz} + \mathbf{qz}$

Choose $z^0 \in R_+^k$, $\omega \in (0,\,2)$. Having z^t compute z^{t+1} as follows

$$\begin{cases} z_j^{t+1} = \Big(z_j^t - \omega M_{jj}^{-1}\Big(\sum\limits_{\substack{\ell=1 \\ \text{for } j>1}}^{j-1} M_{j\ell}\, z_\ell^{t+1} + \sum_{\ell=j}^{k} M_{j\ell} z_\ell^t + q_j\Big)\Big)_+ \\ \qquad\qquad\qquad\qquad j = 1,\ldots,k \end{cases} \tag{3.4}$$

We are ready to state and establish the convergence of our augmented Lagrangian serial SOR algorithm.

3.2 Augmented Lagrangian Serial SOR Algorithm

Let $\{\gamma^i\} \downarrow \bar{\gamma}$ for some $\bar{\gamma} > 0$, let $\{\delta^i\} \downarrow 0$ and let $x^0 \in R^n$. Having x^i determine x^{i+1} as follows:

(a) Apply the Serial SOR Algorithm 3.1 to solve (2.9a) with the identifications (3.2), and let $\big(u^i(t), v^i(t)\big)$ be the t iterate of this SOR algorithm. Stop if for some $t = t^i$ the following inequality is satisfied.

$$\begin{aligned}(3.5)\qquad &\Big|u^i(t)\nabla_u L\big(u^i(t), v^i(t), x^i, \gamma^i\big)\Big| + \Big|v^i(t)\nabla_v L\big(u^i(t), v^i(t), x^i, \gamma^i\big)\Big| \\ &+ \Big\|\big(\nabla_u L\big(u^i(t), v^i(t), x^i, \gamma^i\big)_+\Big\| + \Big\|\Big(\nabla_v L\big(u^i(t), v^i(t), x^i, \gamma^i\big)\Big)_+\Big\| \le \delta^i\end{aligned}$$

(b) Set $x^{i+1} = x^i(t^i)$ where

$$(3.6)\qquad x^i(t) := x^i + \frac{1}{\gamma^i}\big(A^T u^i(t) + v^i(t) - c\big)$$

3.3 Augmented Lagrangian Serial SOR Convergence Theorem

Let $\{\gamma^i\} \downarrow \bar{\gamma} > 0$ be a sufficiently rapidly decreasing sequence of positive numbers and $\bar{\gamma}$ sufficiently small. Then either

(a) For some integer k, the sequence $\{x^k(t)\}$ converges to an $\bar{x}^k$ that solves the linear program (2.1), or

(b) For each subsequence of $\big\{\big(x^i, u^i(t^i), v^i(t^i)\big)\big\}$ converging to some $(\bar{x}, \bar{u}, \bar{v})$, the corresponding subsequence $\{x^{i+1} = x^i(t^i)\}$ converges to an $\hat{x}$ such that $\hat{x}$ solves the linear program (2.1). If $\hat{x} = \bar{x}$, then $(\bar{u}, \bar{v})$ solves the dual linear program (2.2).

Proof Either the inequality (3.5) of the algorithm is satisfied at each iteration i for some t^i or not. Accordingly we have the alternatives (b) and (a) below respectively.

(a) For some iteration $i = k$ the inequality (3.5) is never satisfied. Hence by Pang's Theorem 3.1 [16], since $L(u, v, x^k, \gamma^k)$ is by duality theory bounded above for $(u, v) \ge 0$ by $cx + \gamma^k\|x - x^k\|^2$ for any $x \ge 0$ such that $Ax \ge b$, it follows that the sequence

$$(3.7)\qquad \{x^k(t)\} := \left\{x^k + \frac{A^T u^k(t) + v^k(t) - c}{\gamma^k}\right\},\ t = 0, 1, 2, \ldots,$$

where t is the SOR iteration index, converges to a vector $\bar{x}^k$ defined by

$$\bar{x}^k := x^k + \frac{A^T\bar{u}^k + \bar{v}^k - c}{\gamma^k} \tag{3.8}$$

for some $(\bar{u}^k, \bar{v}^k)$ which solves $\max_{(u,v)\geq 0} L(u, v, x^k, \gamma^k)$. Note however that $\{u^k(t), v^k(t)\}$ need not converge to $(\bar{u}^k, \bar{v}^k)$. If γ^k is sufficiently small (and this is what is meant by requiring that $\{\gamma^i\}$ decreases sufficiently rapidly) it follows by Theorem 2.2 that $x^{k+1} = \bar{x}^k$, is a solution of the primal linear program (2.1). (Note that $(\bar{u}^k, \bar{v}^k)$ need not be a solution of the dual linear program (2.2). To obtain such a dual optimal $(\tilde{u}^k, \tilde{v}^k)$ we need to solve $\min_{(u,v)\geq 0} L((u, v, \bar{x}^k, \gamma^k)$ exactly. See Corollary 3.5 below.)

(b) If inequality (3.5) holds for each i for some t^i, then since $\{\delta^i\} \downarrow 0$ and $\{\gamma^i\} \downarrow \bar{\gamma} > 0$, we have that for any subsequence of $\{(x^i, u^i(t^i), v^i(t^i))\}$ converging to some $(\bar{x}, \bar{u}, \bar{v})$, the point $(\bar{u}, \bar{v})$ solves the problem $\min_{(u,v)\geq 0} L(u, v, \bar{x}, \bar{\gamma})$ (because $\{\delta^i\} \downarrow 0$), and moreover the corresponding subsequence $\{x^{i+1}\}$ defined by (3.6) and part (b) of Algorithm 3.2 converges to an $\hat{x}$ defined by

$$\hat{x} := \bar{x} + \frac{A^T\bar{u} + \bar{v} - c}{\bar{\gamma}} \tag{3.9}$$

If $\bar{\gamma}$ is sufficiently small, then it follows by Theorem 2.2 that $\hat{x}$ solves the linear program (2.1). If in addition $\hat{x} = \bar{x}$, then $(\bar{u}, \bar{v})$ is feasible for the dual linear program (2.2) and is also optimal because $b\bar{u} = c\hat{x}$. ∎

It is useful to point out that when the sequence $\{(u^i(t), v^i(t))\}$ of Algorithm 3.2 is bounded then it has an accumulation point and by Theorem 2.1 [7], each such accumulation point satisfies the optimality conditions for (2.9a). Consequently for each i inequality (3.5) of Algorithm 3.2 is satisfied after a finite number of steps of part (a) of the algorithm. However by Theorem 2(ii) of [9] we have that if the linear program (2.1) satisfies the Slater constraint qualification, then the sequence $\{(u^i(t), v^i(t))\}$ is indeed bounded. Therefore we have the following.

3.4 Augmented Lagrangian Serial SOR Convergence Corollary

Let the linear program (2.1) satisfy a Slater constraint qualification, that is $Ax > b$ for some $x \geq 0$. Then Theorem 3.3 holds with outcome (b).

Another useful observation follows from the fact [18, Lemma 1], [19] that minimizing the augmented Lagrangian $L(u,v,x,\gamma)$ with an optimal value of x and any $\gamma > 0$ gives a solution to the dual linear program (2.2). Hence we have the following.

3.5 Dual LP SOR Solution Corollary

Under the assumptions of Theorem 3.3 a solution to the dual linear program (2.2) can be obtained for either outcome (a) or (b) of Theorem 3.3 by solving respectively

(a) $\min\limits_{(u,v)\geq 0} L(u,v,\bar{x}^k,\gamma^k)$

or

(b) $\min\limits_{(u,v)\geq 0} L(u,v,\hat{x},\bar{\gamma})$

We note immediately that we have left open the procedure by which the sequence $\{\gamma^i\}$ is decreased. This is an inherent theoretical difficulty that arises when using inexact minimization in the subproblems of proximal point or augmented Lagrangian algorithms. Thus the approximate minimization criteria of [21, Criteria A, B, A', B'] are not implementable for our problem, while the assumptions of [2, Section 2.5] are not verifiable for our problem. Computationally we have overcome this difficulty by using the following scheme, often used for updating the penalty parameter in augmented Lagrangian algorithm

$$\gamma^{i+1} = \begin{cases} \gamma^i \text{ if } \|x^{i+1} - x^i\| \leq \mu \|x^i - x^{i-1}\|, \ 0 < \mu < 1 \\ \nu\gamma^i \text{ otherwise, } \ 0 < \nu < 1 \end{cases} \tag{3.10}$$

This scheme works effectively for the solution of very large sparse linear programs as our computational results indicate.

4. Parallel Successive Overrelaxation Algorithm

The key to our parallel SOR algorithm is the use of the parallel gradient projection SOR (GP-SOR) that we proposed in [13] for the solution of (3.1) and which we outline below. Partition the matrix M of (3.1) into r contiguous horizontal blocks as follows:

$$M =: \begin{bmatrix} M_{I_1} \\ M_{I_2} \\ \vdots \\ M_{I_r} \end{bmatrix} \tag{4.1}$$

where the blocks M_{I_j} correspond to the variables z_{I_j}, and $\{I_1, I_2, \ldots, I_r\}$ is a consecutive partition of $\{1, 2, \ldots, k\}$. Now partition M_{I_j} as follows

$$M_{I_j} =: [M_{I_j I_j} \quad M_{I_j \bar{I}_j}] \tag{4.2}$$

where $\bar{I}_j$ is the complement of I_j in $\{1, 2, \ldots, k\}$. Thus $M_{I_j I_j}$ is a principal square submatrix of M with elements M_{st}, $s \in I_j$ and $t \in I_j$. We further partition $M_{I_j I_j}$ into the sum of its strictly lower triangular part $L_{I_j I_j}$, its diagonal part $D_{I_j I_j}$ and its strictly upper triangle part $U_{I_j I_j}$ as follows

$$M_{I_j I_j} =: L_{I_j I_j} + D_{I_j I_j} + U_{I_j I_j} \tag{4.3}$$

Now define a block diagonal matrix K as follows

$$K := \begin{bmatrix} L_{I_1 I_1} & & & \\ & L_{I_2 I_2} & & \\ & & \ddots & \\ & & & L_{I_r I_r} \end{bmatrix} \tag{4.4}$$

where each $L_{I_j I_j}$ is the strictly lower triangular part of $M_{I_j I_j}$. An SOR algorithm can now be performed for each row block I_j, $j = 1, \ldots, r$, simultaneously, that is in parallel. Note that the block diagonal matrix K replaces the traditional strictly lower triangular matrix of the serial SOR. Specifically we have the following.

4.1 Parallel GP-SOR Algorithm for (3.1)

Let $\{I_1, I_2, \ldots, I_r\}$ be a consecutive partition of $\{1, 2, \ldots, k\}$, let E be a positive diagonal matrix in $R^{k \times k}$ and let $z^0 \geq 0$. For $i = 0, 1, 2, \ldots,$ do the following

Direction Generation Define the direction

$$d^i := p(z^i) - z^i := \begin{pmatrix} p_{I_1}(z^i) - z^i_{I_1} \\ \vdots \\ p_{I_r}(z^i) - z^i_{I_r} \end{pmatrix} \tag{4.5}$$

such that $p(z^i)$ satisfies

$$p_{I_j}(z^i) = \left(z^i_{I_j} - \omega E_{I_j I_j}\left(M_{I_j} z^i + q_{I_j} + L_{I_j I_j}(p_{I_j}(z^i) - z^i_{I_j})\right)\right)_+, \; j = 1, \ldots, r \tag{4.6}$$

where $\omega > 0$ is chosen such that for some $\nu > 0$

$$z_{I_j}\left((\omega E_{I_j I_j})^{-1} + L_{I_j I_j}\right) z_{I_j} \geq \nu \|z_{I_j}\|^2, \; \forall z_{I_j}, \; j = 1, \ldots, r \tag{4.7}$$

Stop if $d^i = 0$, else continue.

Stepsize Generation $z^{i+1} = z^i + \lambda^i d^i$

where

$$f(z^i + \lambda^i d^i) = \min_{\lambda} \{f(z^i + \lambda d^i) \big| z^i + \lambda d^i \geq 0\} \tag{4.8}$$

4.2 Remark The principal part of this algorithm consists of the direction generation part (4.6), which can be performed in parallel on r processors. Once this is done the stepsize generation (4.8) is performed and the new value z^{i+1} is shared between the r processors.

The following convergence results were derived in [13] for Algorithm 4.1.

4.3 Theorem (Convergence of the Parallel GP-SOR Algorithm) Let M be symmetric and positive semidefinite. Either the sequence $\{z^i\}$ generated by the Parallel GP-SOR Algorithm 4.1 terminates at a solution of (3.1) or each of its accumulation points solves (3.1).

4.4 Corollary (Parallel GP-SOR special cases) Condition 4.7 of Algorithm 4.1 holds under either of the following two assumptions:

$$0 < \omega < \min_{j=1,\ldots,r} \; \min_{i \in I_j} \frac{2}{E_{ii} \sum\limits_{\substack{\ell \in I_j \\ \ell \neq i}} |M_{i\ell}|} \tag{4.9}$$

(4.10) $0 < \omega < 2$, $E = D^{-1}$ and M is positive semidefinite

Our parallel augmented Lagrangian method consists of replacing the Serial SOR Algorithm 3.1 by the Parallel GP-SOR Algorithm 4.1 in Algorithm 3.2 with option (4.10) for the choice of ω and E. We formally state the algorithm below.

4.5 Augmented Lagrangian Parallel SOR Algorithm

Identical to Algorithm 3.2 except that the Serial SOR Algorithm 3.1 in Algorithm 3.2 is replaced by the Parallel GP-SOR Algorithm 4.1 with condition (4.10) above in force.

5. Computational Results

Our algorithms were tested on random linear programs which were generated as follows. First the matrix A of the linear program (2.1) was generated. Each of its nonzero elements was a random number generated by a uniform distribution on the interval [-100, 100]. The number of nonzero elements in each row was in accordance to a prescribed density and the random position of each nonzero element was determined according to a uniform distribution on the column indices of the matrix. Next a primal-dual solution vector $(\bar{x}, \bar{u})$ was generated from a uniform distribution on the interval [0, 10] with 80% of the components being nonzero. Finally the vectors b and c were chosen so that $(\bar{x}, \bar{u})$ is optimal.

In Figure 1 we give a summary of our computational results for 6 problems with the number of constraints varying between 25,000 and 125,000 and the number of variables varying between 100,000 and 500,000. All tests were performed on a MicroVax II with 16 megabytes of memory and an expanded disk swap space. We are not aware of any other linear programming software that can handle problems of the size that we have attempted on a comparable machine. MINOS [15], a state-of-the-art pivotal linear programming package, cannot handle any of the problems listed in Figure 1 because they are too big for the machine memory using the MINOS configuration. The largest problem attempted on the MicroVax II with MINOS was a problem with 5000 constraints, 20,000 variables and a matrix density of 0.2% with about 200,000 nonzero elements. MINOS was used with the standard partial pricing and scaling options. After 3150 iterations and 49 hours 54 minutes of machine time, the point was infeasible and the objective function was in error by 59% of the exact minimum. By comparison our Algorithm 3.2 solved the same problem in 1 hour and 4 seconds with a primal-dual objective function accuracy of 7 figures, and relative accuracy of not less than 10^{-9} as defined in Figure 1.

The Parallel SOR Algorithm 4.5 was implemented on the Sequent Balance 21000, a multiprocessor that incorporates eight NS32032 processors running at 10MHz, each with a floating point unit, memory management and an 8-kbyte cache sharing a global memory via a 32-bit wide pipelined bus. The machine has 8-Mbytes of physical memory. The operating system DYNIX, is a version of Berkeley 4.2 bsd unix. The computational results are depicted in Figures 2 and 3.

Figure 2 shows the total computing time versus number of processors for four different densities: d=1, 2, 7 and 10 percent for a linear program with 1000 constraints and 4000 variables. All problems were solved to a 7-figure accuracy of the primal-dual objective function and relative accuracy better than 10^{-7} as defined in Figure 1. We observe that the optimal number of processors, that is the one that solves the problem in minimum total time, increases with density as expected. This number is 3 for 1% and 2% densities, 6 for 7% density, and 7 or more for 10% density. This means that for denser problems, a larger number of processors is needed in order to arrive at the shortest solution time. This also means that for denser problems, the communication cost does not become a dominant and hence prohibitive factor until a larger number of processors are used.

In Figure 3 we show results for the case with 10,000 constraints and 40,000 variables, with density of 0.35% and about 1,400,000 nonzero elements. To our knowledge this is one of the largest linear programs solved on this relatively modest sized multiprocessor. One of the reasons that we were able to solve larger problems on the MicroVax II, is that the latter had twice the total memory size of the Balance 21000 and furthermore, the MicroVax was essentially a single-user machine dedicated to the serial SOR algorithm. The optimal number of processors for the low-density case shown in Figure 3 is 4. Just as in the cases of Figure 2, the optimal number of processors should increase with problem density.

We conclude with some observations on the speedup efficiency of our Parallel SOR Algorithm 4.5. We define the speedup efficiency $E(r)$ as the ratio of the actual to the theoretical speedup of the algorithm using r processors instead of 1 processor, thus

$$E(r) := \frac{T(1)}{rT(r)} \tag{5.1}$$

where $T(r)$ is the total time for solving a given problem using r parallel processors. Figure 4 shows the speedup efficiencies for a typical case of a linear program with 1000 constraints, 4000 variables and 2% density. The reason why some efficiencies are over 100% was pointed out in [13]. The explanation is that our Parallel SOR Algorithm 4.5 changes with the number of processors used, because the matrix K defined by (4.4) changes with the number of blocks into which M is divided. Thus we are not comparing identical algorithms when we evaluate the ration $T(1)/rT(r)$ of (5.1). Nevertheless the expression is a valid measure of efficiency in the sense of comparing the theoretical reduced time $T(1)/r$ to the observed time $T(r)$ for an algorithm with a variable K that depends on the partition

of M. If the matrix K is held fixed for $r = 1$ and $r > 1$, then we obtain efficiencies for identical algorithms, and they would all be less than 100%. This was demonstrated in [13]. Nevertheless the present efficiencies of over 100% are indeed very encouraging and also give the additional and somewhat surprising result that a serial implementation of our r-block Parallel SOR Algorithm 4.5 on a single machine, will give for some r a better computing time than the single block Serial SOR Algorithm 3.2. For the specific case of Figure 4, a serial implementation of the r-block parallel SOR with r=2, 3, 4 and 5 will be faster than the single block serial SOR.

Prob. No.	No. constr. $m\times 10^{-3}$	No. var. $n\times 10^{-3}$	Nonzero elements per row	Total Nonzero elements $\times 10^{-5}$	Iter	Time hr:min	Fig. Accur. Obj. Func. P	D	Log_{10} Rel. Accur. 1	2	3	4
							(a)	(b)	(c)	(d)	(e)	(f)
1	25	100	20	5	309	2:31	7	7	-7	-10	-9	-10
2	30	120	25	7.5	392	4:34	7	7	-10	-14	-12	-14
3	40	160	20	8	363	4:56	7	7	-9	-11	-7	-8
4	50	200	16	8	525	7:47	7	7	-9	-8	-9	-8
5	100	400	12	12	967	22:35	5	7	-3	-5	-5	-7
6	125	500	9	11.25	3100	71:40	6	7	-3	-5	-5	-7

(a) Number of correct figures in primal objective

(b) Number of correct figures in dual objective

(c) $\|(-Ax+b)_+\|_\infty / \|(b)_+\|_\infty$ (Relative Accuracy)

(d) $\|(A^Tu-c)_+\|_\infty / \|(-c)_+\|_\infty$ (Relative Accuracy)

(e) $|cx-c\bar{x}|/|cx+c\bar{x}|$, $\bar{x}$: exact, x: computed (Relative Accuracy)

(f) $|bu-b\bar{u}|/|bu+b\bar{u}|$, $\bar{u}$: exact, u: computed (Relative Accuracy)

Fig. 1. MICROVAX II: Serial SOR Algorithm 3.2 test results.

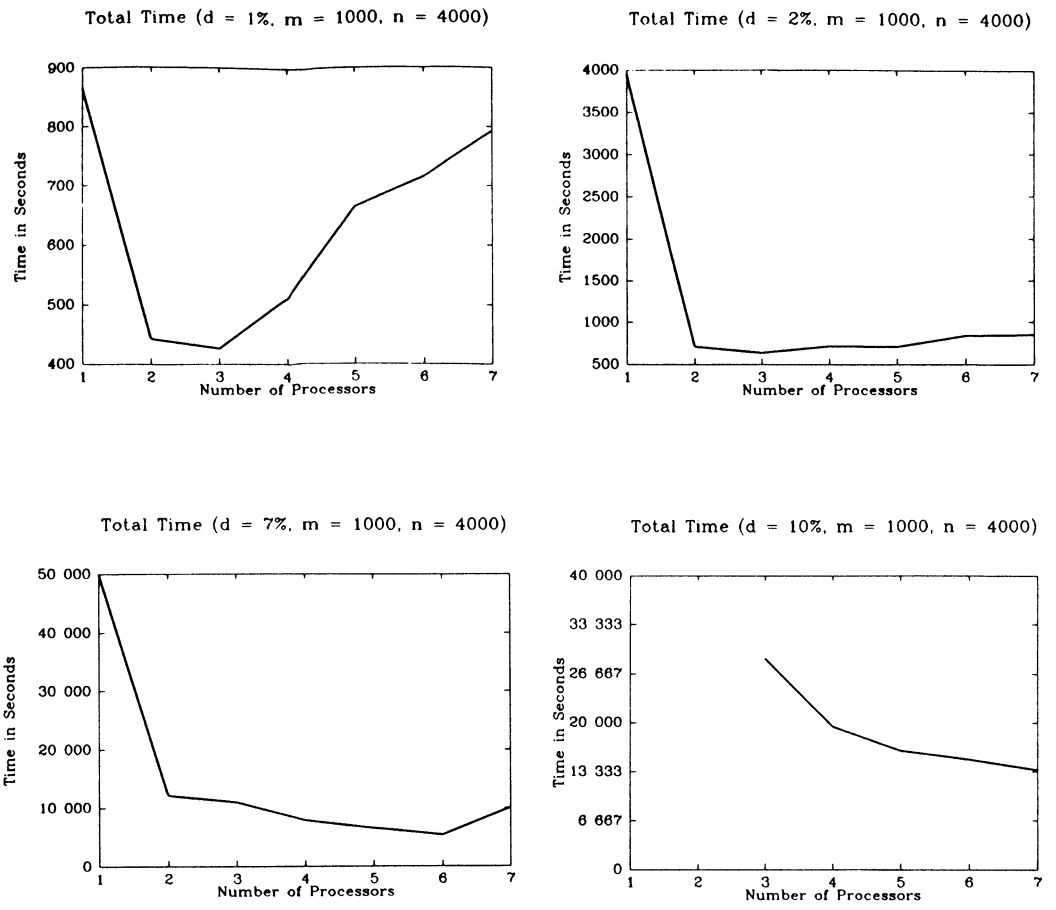


Fig. 2. BALANCE 21000: Total time for Parallel SOR Algorithm 4.5 to solve linear program versus number of processors for various densities d. (Average of 4 randomly generated cases with 1000 constraints and 4000 variables.)

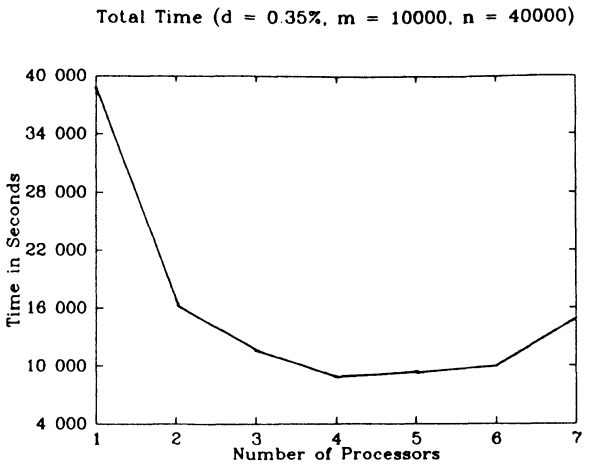


Fig. 3. BALANCE 21000: Total time for Parallel SOR Algorithm 4.5 to solve linear program versus number of processors. (d=density, m=number of constraints, n=number of variables.)

Density $d = 2\%$, $m = 1000$, $n = 4000$		
No. Processes r	Time Sec. $T(r)$	Speedup Efficiency $E(r) = T(1)/rT(r)$
1	3946	—
2	709	278%
3	638	206%
4	716	138%
5	711	111%
6	840	78%
7	850	66%

Fig. 4. BALANCE 21000: Speedup efficiency $E(r)$ for the Parallel SOR Algorithm 4.5 for an LP with 1000 constraints and 4000 variables and 2% density.

References

1. D. P. Bertsekas: "Necessary and sufficient conditions for a penalty method to be exact", Mathematical Programming 9, 1975, 87-99.

2. D. P. Bertsekas: "Constrained optimization and Lagrange multiplier methods", Academic Press, New York 1982.

3. E. G. Golshtein: "An iterative linear programming algorithm based on the use of modified Lagrange function", US-USSR Seminar on Distribution and Production Problems of Industry and Enterprise, October 15-17, 1980, University of Maryland, College Park, Proceedings Published in College Park, Maryland 1982.

4. Y. Y. Lin & J. S. Pang: "Iterative methods for large convex quadratic programs: A survey", SIAM Journal on Control and Optimization 25, 1987, 383-411.

5. O. L. Mangasarian: "Nonlinear programming", McGraw-Hill, New York 1969.

6. O. L. Mangasarian: "Unconstrained Lagrangians in nonlinear programming", SIAM Journal on Control 13, 1975, 772-791.

7. O. L. Mangasarian: "Solution of symmetric linear complementarity problems by iterative methods", Journal of Optimization Theory and Applications 22, 1977, 465-485.

8. O. L. Mangasarian: "Iterative solution of linear programs", SIAM Journal on Numerical Analysis 18, 1981, 606-614.

9. O. L. Mangasarian: "Sparsity-preserving SOR algorithms for separable quadratic and linear programs", Computers and Operations Research 11, 1984, 105-112.

10. O. L. Mangasarian: "Normal solution of linear programs", Mathematical Programming Study 22, 1984, 206-216.

11. O. L. Mangasarian & R. De Leone: "Parallel successive overrelaxation methods for symmetric linear complementarity problems and linear programs", Journal of Optimization Theory and Applications 54(3), September 1987.

12. O. L. Mangasarian & R. De Leone: "Error bounds for strongly convex programs and (super)linearly convergent iterative schemes for the least 2-norm solution of linear programs", Applied Mathematics and Optimization, 1988.

13. O. L. Mangasarian & R. De Leone: "Parallel gradient projection successive overrelaxation for symmetric linear complementarity problems and linear programs", University of Wisconsin Computer Sciences Department Tech. Rept. 659, August 1987, submitted to Annals of Operations Research.

14. O. L. Mangasarian & R. R. Meyer: "Nonlinear perturbation of linear programs", SIAM Journal on Control and Optimization 17, 1979, 745-752.

15. B. A. Murtagh & M. A. Saunders: "MINOS 5.0 user's guide", Stanford University Technical Report SOL 83.20, December 1983.

16. J. S. Pang: "More results on the convergence of iterative methods for the symmetric linear complementarity problem", Journal of Optimization Theory and Applications 49, 1986, 107-134.

17. B. T. Polyak: "Introduction to optimization", Optimization Software, Inc., New York 1987 (translated from Russian).

18. B. T. Polyak & N. V. Tretiyakov: "Concerning an iterative method for linear programming and its economic interpretation", Economics and Mathematical Methods 8(5) 1972, 740-751 (Russian).

19. R. T. Rockafellar: "A dual approach to solving nonlinear programming problems by unconstrained optimization", Mathematical Programming 5, 1973, 354-373.

20. R. T. Rockafellar: "Monotone operators and the proximal point algorithm", SIAM Journal on Control and Optimization 14, 1976, 877-898.

21. R. T. Rockafellar: "Augmented Lagrangians and applications of the proximal point algorithm in convex programming", Mathematics of Operations Research 1, 1976, 97-116.

22. Yu. P. Syrov & Sh. S. Churkreidze: "Questions on the optimization of inter-branch and inter-regional connections in the planning for the growth of a unified national-economic system", Institute of National Economy of Irkutsk, Irkutsk 1970 (Russian).

EMP: An Expert System for Mathematical Programming

- Klaus Schittkowski -

Abstract:
The paper describes a software system that supports model building and numerical solution of various mathematical programming problems, e.g. of nonlinear programming, data fitting, min-max programming, multicriteria optimization, nonsmooth optimization, quadratic programming, or linear programming, subject to linear or nonlinear constraints. All actions of the system are controlled by menues which inform the user how to generate, solve, edit, delete or list a problem, for example. Nonlinear problem functions must be defined by sequences of FORTRAN statements assigning a numerical value to a user-provided name. The system writes a complete FORTRAN source program, which is linked and executed automatically. All problem data and numerical results are stored in a data base and are easily retrieved for further processing. For each nonlinear problem, at least two different mathematical algorithms are available. The system is capable to learn, i.e. to improve its own knowledge on the success of the algorithms, and to perform a rule-based error analysis in case of nonsuccessful termination.

Mathematisches Institut
Universitaet Bayreuth
D - 8700 Bayreuth

1. General information on EMP

The development of efficient and reliable algorithms for solving practical optimization problems was one of the main research areas of the last 20 years in mathematical programming. In the present situation a significant amount of numerical experience and knowledge on the performance of computer programs is available, in particular in linear programming, quadratic programming, least squares optimization, and smooth nonlinear optimization. Only in linear programming, we observe major additional attempts to facilitate model building, program execution and report writing for those who have to solve practical application problems. But based on the previous comment, we have to be aware that also in other areas, mathematical programming codes will be in the hands of non-specialists or occasional users, who have difficulties (or do not possess the time) to find a suitable algorithm, to transform the problem into the required form, to understand the documentation and to execute the FORTRAN code.

Expert systems are computer programs that embody the expertise of specialists in some problem area and were developed in many different disciplines in the last 15 years, e.g. MYCIN for medical consultation, cf. Shortliffe (1976), PROSPECTOR for mineral exploration, cf. Duda, Gasching and Hart (1979), or HEARSAY II for speach recognition, cf. Lesser and Erman (1977). Besides of a suitable knowledge base, an expert system contains an inference mechanism to perform the reasoning process and to construct inference paths. Its basic characteristics can be summarized as follows, see Hayes-Roth, Waterman and Lenat (1982) for additional information:

1. The system must perform well by applying expert rules in an efficient manner to achieve acceptable solutions.

2. Knowledge is to be represented in a symbolic way for realizing the internal reasoning process.

3. The system should possess some knowledge of the basic principles of its domain of interest, e.g. to provide additional explanations when expert rules fail.

4. It must be able to convert the problem description in lay terms into an internal representation convenient for processing with its expert rules.

5. The expert system must be designed to solve domain specific real world problems.

EMP is an interactive programming system that supports model building, numerical solution and data processing of constrained mathematical programming problems. Various options are available in EMP to facilitate the formulation of problem functions. The objective function e.g. may be a linear or quadratic function, a data fitting function, a sum

or maximum of functions, or a general function without a structure that could be exploited. More precisely the following mathematical models are available for facilitating the formulation of objective or constraint functions and exploiting special structures mathematically whenever possible:

- Linear programming
- Linear regression
- Quadratic programming
- Nonlinear L1-data fitting
- Nonlinear L2- or least squares data fitting
- Nonlinear maximum-norm data fitting
- Multicriteria optimization
- Min-max optimization
- Non-smooth optimization
- General nonlinear programming

All problems may have bounds for the variables and linear or nonlinear constraints. In both cases it is possible to proceed from two-sided bounds for the restrictions. Data fitting problems are either composed of a sequence of arbitrary nonlinear functions or of one model function, where the experimental data are provided separately. Further mathematical models are in preparation, e.g. optimal control.

For most optimization problems, many different algorithms are available. The present version of EMP contains mathematical methods of the following type:

- Sequential quadratic programming method for nonlinear programming
- Bundle method for non-smooth or ill-conditioned problems
- Levenberg-Marqurdt, Gauss-Newton, Newton, and quasi-Newton methods for nonlinear least squares problems
- Ellipsoid method for smooth and non-smooth optimization
- Dual and primal methods for quadratic programming

EMP includes program generators for codes of the frequently used optimization libraries NAG and IMSL, and the algorithm base is extended in future step by step.

For objective function and constraints, the input of quadratic or linear functions reduces to definition of some vectors and matrices, respectively, where sparsity can be exploited. Gradients of nonlinear and nonquadratic functions are approximated numerically, but can also be provided by the user in analytical form. The input of 'simple' objective or constraint functions is facilitated, if they differ at most by an index.

Only the problem relevant data need to be provided by a user in an interactive way. General functions must be defined by sequences of FORTRAN statements addressing a numerical value to a user provided function name. All generated problems are stored in form of a data base system, so that they are easily retrieved, modified, or deleted on request. EMP proposes a suitable mathematical algorithm and writes a complete FORTRAN source program. The system executes this

program and stores the numerical results in the data base, so that they are available for further processing. Since individual names for functions and variables can be provided by a user, it is possible to get a problem dependable output of the achieved solution.

The user will be asked whether he wants to link the generated FORTRAN program with some of his own files or whether he wants to insert additional subroutines, declaration and executable statements to formulate the problem. It is possible to generate the same programs automatically, that must be generated by 'hand' otherwise.

In various ways the system is capable to learn and to store its own experience on the success of solution attempts. The proposals offered by EMP, will therefore become better and better with increasing knowledge on the numerical structure of the user provided models. A rule-based failure analysis explains some reasons for possible false terminations and proposes remedies to overcome numerical difficulties.

All actions of EMP are controlled by self-explained commands which are displayed in form of menues. Step by step the user will be informed how to supply new data. Whenever problem data are generated or altered, the corresponding information will be saved on a user provided file. Besides commands to generate, solve or edit a problem, there are others to transfer data from one problem to another, to delete a problem, to sort problems, to get a report on problem or solution data, to halt the system and to get some information on the system, the mathematical models and the available algorithms. It is even possible to insert arbitrary operating system commands without leaving EMP.

The main intention of EMP is to prevent the organisational 'ballast' otherwise required to solve a nonlinear programming problem with a special algorithm. Once the system is implemented, it is not necessary

- to define a special file for each problem to be solved,
- to select a suitable mathematical algorithm,
- to read any documentation of the used mathematical programming algorithms,
- to write long lists of declaration statements, e.g. for dimensioning auxiliary arrays required by the algorithm, or to call the mathematical programming code with a long list of parameters that are to be defined initially,
- to provide the problem functions and their gradients in a special form required by the mathematical programming algorithm,
- to define an individual output file for each set of results,
- to interprete the results for a decision maker.

Thus the domain of application of EMP is summarized as follows:

(i) Programming neighbourhood for developing a first executable program version solving a specific practical problem (or class of problems).
(ii) Investigation of different model variants fitting best to a given real world situation.
(iii) Testing certain types or modifications of mathematical programming algorithms for solving a class of problems.
(iv) Collecting numerical experience on solution methods for optimization problems.
(v) Teaching students on model building (e.g. structural optimization courses in engineering science) or on numerical behavior of optimization algorithms (e.g. optimization courses in mathematics).

EMP allows a user to concentrate all his efforts on the problem he wants to solve and takes over the additional work to select a suitable algorithm and to organize the data, the execution of the problem functions and the program structure. It should be possible to solve optimization problems of the class under consideration within a minimal fraction of time needed otherwise.

EMP is implemented in SUSY, cf. Schittkowski (1987), a language that was designed entirely for the generation of interactive problem solving systems like EMP with the additional feature to process heuristic knowledge and uncertainties. A corresponding interpreter was implemented by the author and is available on request together with EMP, at present only for VAX/VMS computing environments. A UNIX-version is in preparation.

The presented system to facilitate model building and numerical solutionof mathematical programming problems, is based on the host language FORTRAN and it is assumed, that a user is familar with this language. A completely different approach is obtained by designing a specific model language as known from linear programming. An extension to nonlinear problems was implemented by Brooke, Drud, and Meeraus (1984) at the World Bank, and the resulting system is used at present to develop and solve operations research models. However a model language must be learned and the application is restricted to those models that fit into the system. EMP was designed mainly for engineering applications, where we need a very flexible input procedure, since possible application problems are highly nonlinear, and where we can suppose that users are familar with FORTRAN. Since the generated programs can be used outside of the system, one could consider EMP also as a software development tool.

Although EMP can be used in form of a 'stand-alone' product, additional efforts can be undertaken to extend its knowledge base and to improve its intelligence. In particular a user could alter existing models and to define new ones. In addition the system is implemented in a way to allow an extension of the existing algorithm base very easily, i.e. to scratch or add new algorithms for an existing model. The

usage of the special system language SUSY allows an efficient alteration and adaption of the whole program to another situation. A complete documentation of SUSY is included in the system and it is possible to extract individual chapters interactively.

2. Mathematical models

EMP is intended to formulate and solve nonlinear programming problems of the following general structure:

```
minimize        F(x)
subject to      G(1,x)          =  0
                          . . .
                G(me,x)         =  0
                G(me+1,x)       >= 0
                          . . .
                G(m,x)          >= 0
                xl  <=  x  <=  xu
```

The used standard notation can be replaced by actual names for problem functions or variables and has the following meaning:

- n: Number of variables
- me: Number of equality constraints
- m: Number of constraints without lower and upper bounds on the variables
- x: Vector of n independent design variables.
- F(x): Real-valued objective function which is to be minimized. It is assumed that F depends on the n variables x(1), ... ,x(n) and that F is continuous at least in the rectangle defined by the lower and upper bounds on the variables, or is composed of continuous functions.
- G(1), ... ,G(m): Set of m real-valued constraint functions defining the feasible region of the problem. They are assumed to be continuous with respect to the variables x(1), ... ,x(n) at least in the rectangle definied by the bounds on the variables.
- xl,xu: Two n-dimensional vectors defining lower and upper bounds for the variables.

Alternatively it is possible to proceed from two-sided bounds for the constraints, i.e. from a set of constraints of the form

```
                    G(1,x)        =    b(1)   ,
                      . . .
                    G(me,x)       =    b(me)  ,
b(me+1)     <=      G(me+1,x)     <=   b(m+1)  ,
                      . . .
b(m)        <=      G(m,x)        <=   b(m+m-me)  .
```

EMP transforms the above restrictions into a suitable form required by the numerical algorithms. The constant bound b is defined separately from the m constraint functions G(1,x),...,G(m,x).

EMP is also capable to solve linear and nonlinear systems determined by equality and inequality conditions, respectively. In this case, only the constraints are declared by the user and EMP inserts the objective function

$$F(x) = x'*x \ ,$$

so that a feasible solution with minimal Euclidean norm is calculated. Note that x' denotes the transpose of the vector x.

To facilitate input and output of problem functions and to exploit special mathematical structures by the choice of a suitable algorithm, some options are available for providing the functions F and G(1), ... ,G(m).

a) Objective function:

Several options are available for formulating the objective function and for facilitating its input:

- Linear function, i.e.

  $$F(x) = C'*x$$

  with an n - vector C. Only the coefficients of C need to be determined.

- Linear least squares function, i.e.

  $$F(x) = .5*||C*x - D||$$

  where || . || denotes the Euclidean norm. C is an l by n - matrix, D an l - vector. Only C and D need to be defined in this case. Note that all displayed objective function values show the squared norm without the constant term .5*d'*d.

- Quadratic function, i.e.

  $$F(x) = .5*x'*C*x + x'*D$$

  where C is an n by n - matrix, and D an n - vector. Only C and D must be defined in this case.

- L1-data fitting function, i.e.

  $$F(x) = |H1(x)| + ... + |Hl(x)|$$

  The individual functions H1,...,Hl are assumed to be continuously differentiable. Alternatively each function can be considered as a result of a parameter estimation experiment subject to a model function H(t,x), so that

  $$Hi(x) = w(i)*(H(t(i),x) - y(i))$$

for i=1,...,l, where w(i) denotes the weight, t(i) the independent model data, e.g. observation time, and y(i) the dependent or observed data of the i-th experiment, i=1,...,l. It is allowed that t defines a vector of different independent model data. Besides of these data, only the model function H(t,x) needs to be declared, which depends on a real variable t and a vector x of the n independent variables. H should be continuously differentiable.

- L2-data fitting or nonlinear least squares function, i.e.

$$F(x) = H1(x)**2 + \ldots + Hl(x)**2 \quad .$$

The l observation functions H1, ... ,Hl are continuously differentiable and are declared individually instead of the composed function F. As above each Hi can be replaced by a parameter estimation experiment, i.e. by a term of the form w(i)*(H(t(i),x) - y(i)).

- Maximum-norm data fitting, i.e.

$$F(x) = \text{maximum } \{|H1(x)|,\ldots,|Hl(x)|\} \ ,$$

where all individual functions are continuously differentiable. Again each function can be considered as the result of a parameter estimation experiment subject to a model function H(t,x).

- Maximum of functions, i.e.

$$F(x) = \text{maximum } \{H1(x),\ldots,Hl(x)\} \quad .$$

The individual functions H1,...,Hl are continuously differentiable and could be realizations of a parameter estimation experiment subject to a model function H(t,x).

- Vector of functions, i.e.

$$F(x) = (H1(x), \ldots , Hl(x))' \quad .$$

Each individual function must be continuously differentiable. In an interactive way, the user will be asked to define the way how the multicriteria decision is to be performed. Four different attempts are available:

1. Sum of weighted functions
2. Sum of weighted functions with additional bounds on objective functions
3. Selecting one function, suppressing the other ones
4. Combination of ideal solution and the center of best value points

The corresponding weights are definied interactively when executing the FORTRAN-code, and can be modified during the same session.

- Sum of functions, i.e.

$$F(x) = H1(x) + \ldots + H1(x)$$

As above, only the functions H1, ... ,H1 are to be defined by the user and these functions should be continuously differentiable.

- Sum of simple functions, i.e.

$$F(x) = H(1,x) + \ldots + H(1,x)$$

Here it is assumed that each individual function H(i,x), i=1,...,l, possesses the same mathematical structure and depends only on an index i and the vector x of the decision variables. Again the function H(i,x) is to be continuously differentiable.

- General function, i.e. it is supposed that no specific structure is available that could be exploited.

b) Constraints:

The restrictions are either assumed to be linear or of a general nonlinear type:

- No constraints, i.e. m is equal to zero. Since there is no need to provide special purpose algorithms in this case, the resulting problems are solved by the available constrained mathematical programming algorithms.

- Linear constraints, i.e.

$$G(j,x) = x'*a(j) + b(j)$$

The n - vector a(j) is the j-th row of an m by n - matrix A and b(j) the j-th coefficient of an n - vector B, j=1,...,m. Only the data A and B are requested by EMP. If, on the other hand, two-sided bounds are to be used to define the constraints, then linear constraints are formulated in the way

$$\begin{array}{lcll} & x'*a(1) & = & b(1) \quad , \\ & \vdots & & \\ & x'*a(me) & = & b(me) \quad , \\ b(me+1) \leq & x'*a(me+1) & \leq & b(m+1) \quad , \\ & \vdots & & \\ b(m) \leq & x'*a(m) & \leq & b(m+m-me) \quad . \end{array}$$

- General constraints, i.e. at least one of the constraints is nonlinear. It is possible in practice, that all restrictions possess the same structure and depend at most on an index 'i' for the i-th constraint. In this case only some FORTRAN statements for defining G(i,x) must be provided by a user.

The mathematical programming algorithms require the evaluation of gradients of all problem functions. If they cannot be formalated automatically by EMP, i.e. if the problem functions are neither linear nor quadratic, then the user has two options. In the first case, he requires the numerical evaluation by scaled forward differences, and no additional input is necessary. In the second case, he prefers to evaluate them analytically by a sequence of suitable FORTRAN statements. In general it can be recommended to use the numerical approximation option. Analytical derivatives should be determined only if difference quotients produce significant round-off errors or if the additionally required n function evaluations are too expensive compared with one gradient evaluation.

An additional option of EMP is the possibility to perform a statistical analysis, if the objective function describes a parameter estimation problem subject to the L2-norm. In this case the following quantities are evaluated and displayed on request:

- Variance/covariance matrix of the problem data
- Correlation matrix of the problem data
- Standard deviation for the previously calculated parameters
- Estimated variance of the residuals
- Confidence intervals for the individual parameters subject to the significance levels 1,5, and 10 per cent

Note that it is always allowed to direct the output of the optimization routines and the mentioned statistical analysis to a user provided file.

3. Algorithms for solving mathematical programming problems

Depending on the type of the problem functions, EMP creates either a general nonlinear programming, a nonlinear data fitting, a multicriteria optimization, a quadratic programming or a linear programming problem. In all cases, the chosen algorithm is capable to take into account two-sided bounds for the constraints and lower and upper bounds on the variables after a suitable reformulation of the model, if necessary. The linear and quadratic programming problems are formulated only when all constraints are linear.

The following FORTRAN algorithms are available to solve any of the mentioned mathematical programming problems. Detailed user documentations are contained in EMP and can be displayed interactively.

a) NLPQL (Schittkowski (1985/86)):

NLPQL realizes a sequential quadratic programming (SQP-) method for solving the general nonlinear programming problem with continuously differentiable objective and constraint functions. The algorithm formulates a sequence of quadratic programming problems by linearizing the constraints and approximating the Lagrange function quadratically. By solving these subproblems, a search direction for the variables and the multipliers is obtained. Subsequently a line search is performed along that direction, to get a new iterate, i.e. a new guess for the variables and multipliers. The merit function to calculate the steplength parameter, is an augmented Lagrangian or an L1-exact penalty function alternatively. Then a new approximation matrix for the Hessian of the Lagrange function is updated by the BFGS-quasi-Newton formula.

The SQP-method was first proposed by Wilson (1963) and investigated by Han (1976,1977) and Powell (1978a,1978b) in more detail. In particular they proved that the algorithm converges superlinearly when starting sufficiently close to an optimal solution. The mathematical algorithm of the used implementation is described completely in Schittkowski (1983) together with a global convergence result. This theorem states that starting from an arbitrary initial point, a Kuhn-Tucker point of the problem will be approximated, that is a point satisfying the necessary optimality conditions.

For the problem type under consideration, i.e. for smooth, small scale and eventually highly nonlinearly constrained problems, the SQP-method can be considered as the most powerful solution technique, see Schittkowski (1980) or Hock, Schittkowski (1983) for an extensive general purpose comparison with other available methods, or Schittkowski (1984/86) for performance results of NLPQL. Despite of its excellent average efficiency and reliability, an SQP-

algorithm can fail under certain circumstances, e.g. if
- the problem functions are not continuously differentiable or even not continuous in the feasible region defined by the lower and upper bounds on the variables,
- the problem functions can not be evaluated within a sufficiently small accuracy,
- the problem is badly scaled, ill-conditioned or does not satisfy the constraint qualification, i.e. the linear independency assumption of gradients of active constraints,
- the starting point is too far away from a solution, so that the solution progress is too small,
- the problem possesses several local solutions and the approximated one is not appropriate.

Typical reaction of an SQP-method in one of the above situations, is a slow convergence indicated by attaining the maximum number of iterations, or the inability to find a suitable steplength. In such a case, EMP allows to choose other solution methods to overcome the mentioned difficulties.

NLPQL is now availble in the IMSL-library under the names NCONF and NCONG.

b) E04VDF (NAG: Gill e.al. (1983a)):

The underlying mathematical idea is very similar to that in NLPQL, i.e. E04VDF realizes a sequential quadratic programming method. In contrast to NLPQL, the Hessian of an augmented Lagrangian is approximated instead of the Lagrangian itself, and a 'hot' start is possible for the quadratic programming subproblem, to get an initial guess for the first basis. It is allowed to define linear constraints separately and all constraints possess upper and lower bounds.

EMP generates the same FORTAN subroutines NLFUNC and NLGRAD, which are used to solve the problem by NLPQL. Safeguards prevent unnecessary evaluations of function values. If two-sided bounds for constraints are not defined by the user initially, reasonably large values are inserted.

c) ELL (Schittkowski (1985)):

The algorithm ELL belongs to a class of methods, that do not depend on
- an initial guess for the variables,
- any multiplier approximations,
- scaling properties of the problem functions,
- differentiability assumptions.

In this case, it will be tried to solve the nonlinear programming problem by a variant of the ellipsoid-method, which was proposed by Shor (1977) for solving convex mathematical programming problems. Starting with an ellipsoid

containing the rectangle defined by lower and upper bounds on the variables, a sequence of ellipsoids is constructed with volumes decreasing linearly to zero. Any new ellipsoid is uniquely determined by the requirement, that it should contain one half of the previous one with minimal volume. The cutting half-space is determined by the gradient of a special penalty function. Iterates are the centers of the ellipsoids.

Originally, an ellipsoid method is only capable to solve unconstrained problems. Constraints and bounds for the variables are taken into account by an L1-penalty function. The corresponding penalty factor must be defined by the user.

The convergence speed of the ellipsoid method is at most linear and depends on the number of variables. Since multipliers are not approximated, a precise stopping condition cannot be given in the constrained case. However it is possible to predetermine the number of iterations to achieve a relative reduction of the volumes with respect to the starting ellipsoid. An additional termination parameter is the desired accuracy in constraint satisfaction. The default value is three digits and should not be increased too large.

Theoretically, the method converges only if the underlying problem is convex. But the numerical results of Ecker and Kupferschmid (1983) show that the algorithm is quite robust also in the non-convex case. From the numerical point of view, the method is in general less efficient than an SQP-method, i.e. uses a larger number of function and gradient evaluations in particular when a high accuracy is required. But it can be used quite sucessfully to obtain a suitable starting point or to 'move away' from an undesired local solution.

d) M1FC1 (Lemarechal, Strodiot, Bihain (1981)):

The algorithm solves unconstrained nonlinear programming problems either with a smooth or nonsmooth objective function. Constraints are taken into account by formulating an L1-penalty function, where the corresponding penalty parameter must be determined by the user. If a gradient of the objective function or a constraint is to be calculated analytically at a non-differentiable point, then at least a subgradient must be available.

Proceeding now from an unconstrained optimization problem, the algorithm collects a certain user provided number of previously computed gradients to formulate a bundle, i.e. an interior approximation of the eps-subdifferential. A minimum-norm feasible point of the bundle is calculated then to get a search direction. Subsequently a line search is performed to obtain a new iterate.

Besides of the penalty parameter and the number of

gradients determining the bundle, a user must specify several stopping tolerances and in particular a numerical figure that determines the expected decrease of the objective or penalty function, respectively. Numerical tests indicate that this number should be chosen very carefully, since the performance is sensitive to its choice.

e) ZXMWD (IMSL: Fletcher (1972)):

Problems with box constraints are transformed into unconstrained problems by using trigonometric functions. General nonlinear constraints are taken into account by an L2-penalty function. The unconstrained problem is then solved by a quasi-Newton method of Fletcher (1972). Smooth problem functions are required.

Besides of the penalty parameter, the user is asked to specify a certain number of starting points which are generated randomly. Then about four iterations are performed by the quasi-Newton method and the five results with the lowest function values are then taken to continue the iteration. Thus the code can be used in particular when it is supposed that the problem possesses various local minima.

f) MCO (Schittkowski (1986b)):

The multicriteria problem mentioned in the previous section, is transformed into a nonlinear programming problem with one differentiable objective function, which is then solved by NLPQL. Four different attempts are available to obtain the transformed problem: (1) Sum of weighted functions
(2) Sum of weighted functions with additional bounds on objective functions
(3) Selecting one function, suppressing the other ones
(4) Combination of ideal solution and the center of best value points
The corresponding weights are defined interactively when executing the FORTRAN-code, and can be modified during the same session. A list of efficient points calculated during an interactive session, is kept in memory for further processing. Their number can be determined by the user among some other stopping tolerances used for NLPQL.

g) DFNLP (Schittkowski (1985c)):

Although many efficient special purpose algorithms were developed in the past to solve data fitting problems, only very few codes are available that are able to take into account nonlinear constraints. The program DFNLP solves constrained data fitting problems in L1, L2 and maximum norm and, in addition, min-max problems by transforming them into a suitable nonlinear programming problem. This problem is

then solved by algorithm NLPQL.

(i) L1-data fitting problem: We introduce l additional variables z(1),...,z(l) to transform the original problem into a nonlinear programming problem with objective function

$$F(x,z) = z(1) + \ldots + z(l)$$

and l+l additional inequality constraints of the form

$$-Hj(x) + z(j) >= 0 \quad , \quad j=1,\ldots,l \ ,$$
$$Hj(x) + z(j) >= 0 \quad , \quad j=1,\ldots,l.$$

Obviously both problems are equivalent.

(ii) L2-data fitting or nonlinear least squares problem: By introducing l additional variables z(1), ... ,z(l), the least squares problem is transformed into a general nonlinear programming problem with objective function

$$F(x,z) = z(1)**2 + \ldots + z(l)**2$$

and l additional equality constraints

$$Hj(x) - z(j) = 0$$

j=1,...,l. It turns out that an SQP-method applied to the modified problem, retains some characteristic features of a special purpose least squares algorithm, in particular the usage of a Gauss-Newton direction combined with a quasi-Newton approach. More details and some computational test results are found in Schittkowski (1985). EMP will ask the user whether he has some impression on the size of the residual. If he expects a small residual problem, then it will be tried to reduce the influence of the quasi-Newton matrix as much as possible. Moreover he should consider the possibility to restart the solution algorithm by defining a low maximum number of iterations, and by solving the problem again with an initial point which is known from the previous run. The resulting numerical algorithm takes advantage of the special structure of the quadratic programming subproblems, so that especially problems with a large number of experiments do not lead to excessive calculation times.

(iii) Maximum-norm data fitting: In this case only one additional variable z is required to transform the problem into a smooth nonlinear programming problem with objective function

$$F(x,z) = z$$

and l+l additional constraints

$$-Hj(x) + z >= 0 , \quad j=1,\ldots,l,$$
$$Hj(x) + z >= 0 , \quad j=1,\ldots,l.$$

(iv) Min-max problem: Very similar to the above case, one additional variable z is introduced to obtain an objective function

$$F(x,z) = z$$

and l additional constraints of the form

$$-Hj(x) + z >= 0 , \quad j=1,\ldots,l,$$

to obtain an equivalent smooth nonlinear programming problem.

h) DFELL (Schittkowski (1986d)):

DFELL realizes an ellipsoid algorithm for solving L1, L2, maximum norm and min-max problems directly without transforming them to a constrained nonlinear programming problem as done in DFNLP. Constraints are taken into account by formulating an L2-penalty function. After some arrangements of data, the code ELL is executed to solve the problem.

i) DFM1FC1 (Lemarechal, Strodiot, Bihain (1981)):

Very similar to DFELL, the least squares problem is transformed into an unconstrained problem and then solved by M1FC1.

j) NLSNIP (Lindstroem (1983)):

The algorithm solves nonlinear least squares problems subject to nonlinear equality and inequality constraints. The Gauss-Newton method is modified to handle constraints by an active set strategy. By solving the resulting linear least squares problem with linear equality constraints, a search direction is obtained. Using a penalty function subject to a norm that must be specified by the user, a line search is performed to get a new iterate. To stabilize the algorithm, a subspace minimization option is included in particular for large residual problems. Alternatively it is possible to provide second derivatives and the algorithm switches to Newton's method automatically, if necessary, i.e. if the Gauss-Newton search direction does not lead to a sufficient success.

First derivatives are either defined analytically or are computed numerically by forward differences. If a user prefers the usage of Newton's method, second derivatives are approximated numerically. When starting NLSNIP, the desired

norm of the penalty function used for the line search, the usage of Newton's method and some stopping tolerances must be specified.

k) E04FDF (NAG: Gill, Murray (1978)):

The algorithm solves unconstrained least squares problems by a modified Gauss-Newton method and uses only function evaluations. Gradients are approximated numerically.

Constraints and bounds are not taken into account, and are simply omitted when solving a problem. For the evaluation of the problem functions, EMP generates the same subroutine called DFFUNC which is formulated also for the least squares algorithms DFNLP, DFELL and DFM1FC1. This routine is then called by the special purpose subroutine required by E04FDF.

l) E04GCF (NAG: Gill, Murray (1978)):

The algorithm solves unconstrained least squares problems by a modified Gauss-Newton method and requires also gradient evaluations. A quasi-Newton method is used to approximate part of the Hessian of the objective function.

Constraints and bounds are not taken into account, and are simply omitted when solving a problem. For the evaluation of the problem functions and gradients, EMP generates the same subroutine called DFFUNC and DFGRAD which are formulated also for the least squares algorithms DFNLP, DFELL and DFM1FC1. These routines are then called by the special purpose subroutines required by E04GCF.

m) E04GEF (NAG: Gill, Murray (1978)):

The algorithm solves unconstrained least squares problems by a modified Gauss-Newton method and requires also gradient evaluations. A difference formula is used to approximate part of the Hessian of the objective function.

Constraints and bounds are not taken into account, and are simply omitted when solving a problem. For the evaluation of the problem functions and gradients, EMP generates the same subroutine called DFFUNC and DFGRAD which are formulated also for the least squares algorithms DFNLP, DFELL and DFM1FC1. These routines are then called by the special purpose subroutines required by E04GEF.

n) ZXSSQ (IMSL: Brown, K.G., Dennis, J.E. (1972)):

The unconstrained least squares problem is solved by a modification of the Levenberg-Marquardt algorithm. Gradients are evaluated numerically. Some options are available to

modify the Marquardt parameter before starting the algorithm. Constraints are not taken into account.

o) QL (Schittkowski (1986e), Powell (1983)):

An efficient quadratic programming algorithm is required not only for solving special problems of this type, but also for implementing any SQP-method. In the present case, an algorithm developed by Powell (1983) was modified to satisfy additional requirements, e.g. to handle upper and lower bounds separately or to exploit initially given Cholesky-factorizations. The mathematical method was proposed by Goldfarb and Idnani (1983) and can be characterized as a dual approach. Starting from a Cholesky-decomposition of the objective function matrix, the minimum of the unconstrained problem is found immediately. Successively all violated constraints are incorporated into a 'working set' defining a certain subspace, where the minimum of the objective function is recalculated. It might be necessary to drop constrains from the working set, until the optimal solution is reached.

p) E04NAF (NAG: Gill e.al. (1983b)):

The program solves quadratic programming problems by the primal method of Gill e.al. (1983b), which can be characterized as an active set strategy. The objective function matrix must be symmetric, but is allowed to be indefinite. The algorithm allows a 'warm' start and possesses many additional features, so that the reader is referred either to the original report or the program documentation included in EMP, to get more information.

q) LP (Schittkowski (1986f)):

EMP is not designed to solve large and sparse linear programming problems or linear programming problems with a special structure. A specific modelling language as known from linear programming applications, is therefore not available. The input of sparse matrices is facilitated in the way that the user is allowed to to insert only the non-zero entries of matrices and vectors, or to define the matrix values by one arithmetic expression. In the latter case, the elements may be calculated somewhere else, so that only values of a two-dimensional array can be transferred. The linear program is solved by the simplex method.

r) E04MBF (NAG: Gill e.al. (1983b)):

This code is based on the quadratic programming method E04NAF and uses the primal method included there. Linear constraints must be defined with two-sided bounds.

s) ZX3LP (IMSL)

The code realizes a simplex algorithm. A transformation of the given problem is required, since the code accepts only lower zero bounds for the variables.

Linear least squares problems are transformed into quadratic programming problems. Since the resulting positive semi-definite matrix of the objective function is calculated in double precision arithmetic, the additonally introduced round-off errors might be tolerable.

4. Commands of EMP

All actions of EMP are controlled by commands. They can be displayed in form of a menue by typing the questionmark '?'. From the viewpoint of a user, they should be considered as self-contained programs, i.e. they can be used independently from each other. If problem data are generated or altered, the corresponding data will be saved on a file provided by the user. Whenever EMP executes an operating system command, e.g. for compiling and linking a FORTRAN program, a separate subprocess is generated and executed. Subsequently EMP is continued, but the user will not realize such a situation.

Once a command is specified, EMP asks the user to answer further questions e.g. in the form

* Analytical derivatives (Y/N):

In the above example, a user has to type either 'Y' for yes or 'N' for no. It is always assumed that the less probable answer is 'Y', so that in this case a 'Y' has to be typed exactly and any other input is interpreted as 'N'. But whenever a user does not know the precise meaning of the brief information given, he is allowed to press '?'. He obtains then more detailed information before he has to answer the question again. In the same situation it might happen that a user wants to cancel the present execution of a command, e.g. if he notices a severe previous input error. By typing the sign '<' he will return immediately to main menue.

By the following subtopics, only a rough description of available EMP commands will be given to understand their basic operations. Their particular usage is self-explained by sub-menues, additional information displayed on the terminal whenever it seems to be necessary and the help-option of EMP.

a) INFORMATION:

Displays some information on the screen to understand the system, the mathematical models and algorithms, or the available commands. The displayed text is identical with this documentation which is used to extract the desired sections. Alternatively the whole documentation can be formatted and printed on request by a special command option.

b) GENERATE:

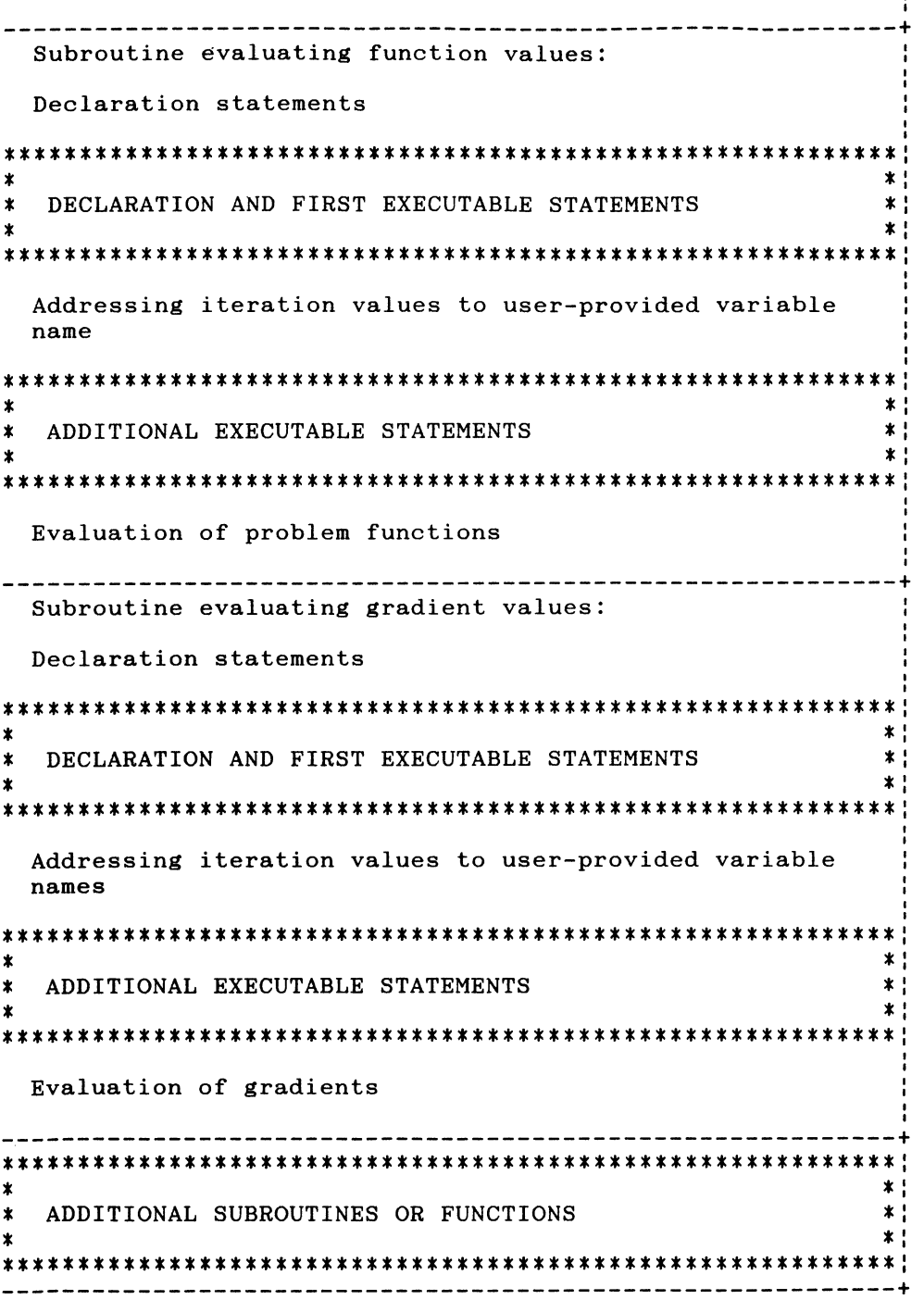
In an interactive way, the user will be informed how to construct a new mathematical optimization problem. After input of the problem name and, optionally, some additional information on the practical background, the type of the problem and its functions must be specified. Then some dimensioning parameters are required, e.g. number of

variables and constraints. User provided declaration statements, constant data, scaling parameters or parameter estimation data follow whenever required for constructing the actual problem under consideration.

Besides the input of the function declaration statements, it is possible to define additional FORTRAN statements which will be inserted either into the main program or the subroutines evaluating the problem functions and gradients. They are placed directly after assigning actual iteration values to the variables, which can therefore be modified or used to prepare some initial data before calling the optimization algorithm or calculating the function and gradient values, respectively. Further statements can be inserted into the main program after executing the optimization subroutine, e.g. to process the achieved results. In this case, the user provided variable and function names contain the computed numerical values. Moreover the user may add some arbitrary FORTRAN subroutines or functions, which will be appended to the generated source code.

The input of additional FORTRAN statements is to be illustrated in form of a sketch. The blocks surrounded by '*' are optional and can be inserted by the user in form of arbitrarily many FORTRAN statements.

```
+-------------------------------------------------------------+
|  Main Program:                                              |
|                                                             |
|  Declaration statements                                     |
|                                                             |
|*************************************************************|
|*                                                           *|
|*  DECLARATION AND FIRST EXECUTABLE STATEMENTS              *|
|*                                                           *|
|*************************************************************|
|                                                             |
|  Preparation of initial data, addressing starting values    |
|  to variables                                               |
|                                                             |
|*************************************************************|
|*                                                           *|
|*  ADDITIONAL EXECUTABLE STATEMENTS                         *|
|*                                                           *|
|*************************************************************|
|                                                             |
|  Call of the mathematical programming algorithm             |
|                                                             |
|*************************************************************|
|*                                                           *|
|*  PROCESSING THE ACHIEVED RESULTS                          *|
|*                                                           *|
|*************************************************************|
|                                                             |
|  Storing results on a file                                  |
```


Subroutine evaluating function values:
Declaration statements
DECLARATION AND FIRST EXECUTABLE STATEMENTS
Addressing iteration values to user-provided variable name
ADDITIONAL EXECUTABLE STATEMENTS
Evaluation of problem functions
Subroutine evaluating gradient values:
Declaration statements
DECLARATION AND FIRST EXECUTABLE STATEMENTS
Addressing iteration values to user-provided variable names
ADDITIONAL EXECUTABLE STATEMENTS
Evaluation of gradients
ADDITIONAL SUBROUTINES OR FUNCTIONS

In the above sketch, it is assumed that the used optimization program requires the evaluation of functions and

gradients in two different subroutines. It is, however, possible that functions and gradients are to be calculated in one subroutine or even in two subroutines separately for objective function(s) and constraints. In this case the additionally defined FORTRAN statements are all inserted into each subroutine that evaluates functions and gradients together.

Except for the quadratic or linear programming problems, it is necessary to provide initial estimates for the variables and to declare upper and lower bounds. If all initial values and bounds are identical or depend at most on an integer 'i', then the input of these data can be reduced to only one line by typing a suitable arithmetic expression depending on an 'i'.

The evaluation of the problem functions depends on their specific type defined earlier. If a function is linear or quadratic for example, then only the corresponding matrices or vectors must be defined. In the general case, the input of those functions is facilitated, which can be evaluated within one line. Otherwise an arbitrary number of FORTRAN statements must be typed by the user for addressing a numerical value to a function, i.e. for assigning a numerical value to a function name.

When defining a problem function by a couple of FORTRAN statements, a user should avoid conflict with existing labels and names. In the first case, the most simple way to prevent an overlapping of labels, is the restriction to numbers between 1 and 1000. The first character of any variable should satisfy the FORTRAN convention, i.e. should be in the set 'a' - 'h', 'o' - 'z'. The numerical algorithms require additional real variables for some internal organisational reasons. But all real variables defined by the system, consist of six characters, so that the easiest way to avoid conflicts, is the usage of real names with less than six signs. Integer values like I, J, K are used by EMP only within 'closed loops', i.e. for control constructs where a user cannot insert his own statements. Therefore they are free for defining function statements. As above, all names of FORTRAN-subroutines reserved by EMP, consist also of six characters and any conflict is avoided by using at most five characters for user provided subroutine or function names. In a similar manner, named COMMON-blocks required by an optimization algorithm, are identified by six signs.

The last action of GENERATE is the request to determine those file names, which contain additional subroutines and which are to be linked with the FORTRAN code generated by EMP, and the corresponding solution algorithms. Note however, that GENERATE does not write the mentioned code, but saves all collected data on a user provided file for further processing.

c) SOLVE:

This commands starts the numerical solution of a problem that was previously constructed by GENERATE. After input of the problem name, a user has the option to require the display of all available codes that could solve his optimization problem. The list contains a certainty factor for each proposed program which indicates a very rough approximation of a measure for the numerical performance of an algorithm. A value of 100 is the maximum attainable degree of belief, whereas a value of 0 indicates a very poor performance of the algorithm on previous runs. The numerical values of the certainty factors are defined and updated in three different ways:

a) Initially every code obtains a certain permanent default value of the system author which is based on his own subjective experience.

b) Whenever a code could solve a problem succcessfully, so that the stopping criteria are satisfied subject to the provided tolerances, the corresponding permanent certainty factor of the code is increased. If, on the other hand, a failure is reported, then the permanent certainty factor is decreased. The factor is not altered at all if the iteration was interrupted due to the fact that the maximum number of iterations was attained.

c) When starting the solution of a problem, a local certainty factor is defined which gets the value of the permanent one, and all previous numerical solution attempts are investigated. If the problem could not be solved by a specific algorithm, then the local certainty factor is decreased significantly, and enlarged otherwise.

The local certainty factors are displayed to support the selection of a suitable code. It is hoped that the choice of these factors reflects the influence of special model structures and that EMP is capable to learn, i.e. to propose better and better solution methods in dependance on the problem to be solved. It is selfevident that a user may reject the algorithm which got the largest certainty value, and to choose another one.

Subsequently some additional decisions must be made by a user, e.g. the choice of a suitable output flag. It is possible that a previously computed and eventually inaccurate approximation of the solution is available. Then the user is asked whether he wants to accept this guess as a starting point for the actual run or not. If some additional output from the underlying mathematical programming algorithm is required, then the information is displayed on the terminal in original form, in particular without individual user-provided names for functions or variables.

The generated FORTRAN code is written on a text file with the name 'EMPCODE.FOR'. The code consists of a main program and, if the problem functions are nonlinear, two subroutines for evaluating problem functions and corresponding gradients in a form required by the selected optimization code. EMP compiles this program, links it with a chosen nonlinear programming algorithm and eventually some user files, and executes the resulting program. All numerical results, performance data and termination reasons are stored automatically in the underlying data base and are available for further processing. Afterwards the main-menue of EMP is displayed again and the user could choose additional actions, e.g. to investigate the obtained results. Note that after leaving the system, the last generated FORTRAN program is still available and could also be used furtheron independently from EMP. It is possible to direct the output of an optimization program to an existing file on request.

d) EDIT:

Command EDIT allows to modify all available problem data and function statements interactively. After defining the desired problem name, the user has to specify whether he prefers the usage of the SUSY editor or the operating system editor. Then a menue allows him to indicate the specific data or functions that are to be edited.

If the SUSY editor is chosen, then a user must know the commands that are allowed to edit lines and which consist of certain signs to be typed at the first position of the actual input line. The following options are available:

A	Add a line (following directly the displayed one).
M	More than one line are to be added.
R	Replace the displayed line completely.
S	Scratch displayed line from the file.
E	Exchange characters of the displayed line by non-blank characters typed in the input line.
I	Insert additional characters at positions after the last blank sign of the input line.
B	Blanks are to be inserted proceding the position indicated by an arbitrary non-blank character in the input line.
D	Delete characters marked be arbitrary non-blank signs in the input line.
N	Next line.
Ni	Proceed to the i-th line following the displayed one.
P	Previous line.
Pi	Proceed to the i-th previous line.
H	Halt editing the actual block of statements.
<	Return to the beginning of the EDIT-command.

e) TRANSFER:

It is sometimes desirable to generate or modify a problem that is very similar to an existing one, e.g. if a new problem is to be solved with the same objective function, but some other constraints or constant data. A transfer procedure allows to transport information from a source problem to a destination problem. If the latter one exists in the problem base, then the corresponding data will be overwritten. Otherwise a new problem is generated with the same problem parameters and the data to be transferred, are inserted.

A user should apply the TRANSFER-command very carefully when the underlying problem structure is to be altered. Particular attention is necessary if the number of variables or functions is different in both problems.

f) DELETE:

Any problem that exists in the data base, can be deleted on request by the user. Unless EMP is not stopped by the HALT-command, a deleted problem is still available on a backup file with name 'EMPBACKUP.DAT', and can be retrieved in an error case. In this case the system should be left by '<ctrl>-Y' and the backup file is copied to the data base file defined initially when starting EMP. If two problems possess the same name, only the first one will be deleted.

g) REPORT:

This command displays problem data and function statements on the one hand and numerical solution data on the other. In both cases it is possible to get a complete list of all available problem names together with some condensed information, i.e. problem parameters or performance results. After typing the problem name, a user is asked to specify the data he wants to see. In principal he will be informed on every parameter, data or statement, that was used to generate the problem.

If the problem was solved numerically, it is possible to get a detailed report on all obtained results. Moreover a complete optimization history can be displayed on request. Then some performance data are shown, e.g. the number of function and gradient evaluations, that were used to obtain the current solution. If a problem could not be solved successfully or if a user is not satisfied by the obtained results, then a failure analysis can be required that shows the user not only possible reasons for failures or unsatisfactory results, but also some remedies to overcome this situation, see the description of the command FAILURE ANALYSIS.

An additional feature of REPORT is the possibility to ask

for a line printer output of all problem data and all numerical results for any problem in the data base. The corresponding data are written on an auxiliary file with name 'EMPDRUCK.DAT' that is not deleted after leaving EMP and that can therefore be used also for other purposes.

h) ASSORT:

Whenever a new problem is generated, it will be placed at the first position of the corresponding problem file or data base, respectively. However it might be desirable to sort the data base to get a more convenient problem listing on the screen when displayed by the REPORT-command. Therefore problems can be sorted with respect to a few possible keys, e.g. problem name, dimension, or problem type. These keys are listed on the screen on request and are initiated by typing a specific character.

i) FAILURE ANALYSIS:

It is quite usual, that a problem could not be solved successfully so that the stopping criteria are satisfied within some given tolerances. All algorithms incorporated in EMP, report error situations by certain flags that are displayed in the optimization history table among other performance data. A value of '0' indicates a successful return as defined above, a value of '1' that the maximum number of iterations was attained, and a value greater than '1' a specific error situation depending on the algorithm chosen. To find out the meaning of the displayed numerical figure, the command FAILURE ANALYSIS can be executed. In general, the corresponding information is taken from the user guides of the corresponding programs.

In some cases, e.g. if the code NLPQL or any other one based on NLPQL is chosen, some heuristic knowledge is available on possible origins of the failure. This knowledge is represented in form of rules with certainty factors and by executing the reasoning process, some possible reasons for the failure can be displayed in the order of their significance. Moreover these reasons are exploited again by rules to propose some alternative remedies. The user could investigate these recommendations and follow one of them whenever it seems to be profitable. An additional option is the possibility to require some explanations for understanding the reasoning process.

A typical rule might be of the form

```
      IF  MUL>1.E+5
      THEN  RSNCQU  WITH  30
```

reading as

```
       'If the multiplier values are very large,
        then there is some evidence that the constraint
        qualification might be violated.'
```

The above rule asserts that a certain consequent is emphasized with a given degree of uncertainty, which could also be supported by other rules. Moreover there are other rules to conclude a possible remedy, e.g.

```
IF   RSNCQU  WITH  50  TO  100
THEN REMNEWIN WITH 95
```

reading as

'If the violated constraint qualification might be a possible reason for false termination, then it is recommended to use a new starting value with a high certainty.'

Consequents must consist of so-called actions, i.e. an arbitrary and eventually empty sequence of SUSY-commands, where antecedents are either logical expressions or again actions which have to be verified up to a certain range of uncertainty in order to be taken into account.

j) OPERATING SYSTEM COMMAND:

When starting EMP, the file containing the physical data base must be read and the corresponding data must be processed in a specific way. A few other data management procedures are invoked by the interpreter, so that some execution time must be spent for each call of EMP depending on the size of the data base. To avoid the waiting time if only a few operating system commands are to be executed, a user may require the direct input of a sequence of operating system commands. By an empty input, the commands are sent to the operating system and executed immediately without leaving EMP. After finishing the last command, the main menue is displayed again.

k) USER PROVIDED SUSY COMMANDS:

Although it is tried to organize EMP in a way that all possible actions can be performed by activating suitable menues, there might be the possibility that a user wants to operate directly on the underlying data base. By choosing this command, he will first be asked whether he wants to be informed on a part of the SUSY documentation and whether he wants to see the data structures defined in the main program of EMP. It is an important assumption to know the names of available variables, in particular of those used for constructing the general data structure of an optimization problem with the name 'prob'. Subsequently any valid SUSY command may be typed which is executed immediately. By an empty input, the interactive definition of SUSY commands is stopped and the main menue is shown again.

Possible reasons for the interactive input of SUSY-commands are sorting of problems by other keys, searching for special strings, or, in particular, the copy of optimization problems from one database to another. In this case, the commands

```
REWRITE AUX
WRITE PROB.. TO AUX
```

write the actual and all following problems of the data base to the auxiliary file AUX. After stopping the system and calling it again now with another physical data base, i.e. another initial file name, the interactive commands

```
RESET AUX
READ PROB.. FROM AUX
```

read all problems from AUX and insert them into the new data base directly after the actual problem.

1) HALT:

By pressing 'H' for activating the command HALT, the system will be left and the user returns to his operating system. The actual data base will be stored on the file provided initially by the user. A backup file with the name 'EMPBACKUP.DAT' is also available containing the last updated problem base.

5. References

The following list shows references of papers that are mentioned in other sections.

Brown, K.M., Dennis, J.E. (1972): Derivative free analogues of the Levenberg-Marquardt and Gauss-Newton algorithms for nonlinear least squares approximations, Numerische Mathematik, Vol.18, 289-297.

Ecker, J.G., Kupferschmid, M. (1983): An ellipsoid algorithm for nonlinear programming, Mathematical Programming, Vol.27, 83-106.

Fletcher, R. (1972): Fortran subroutines for minimization by quasi-Newton methods, Report R7125 AERE, Harwell, England.

Goldfarb, D., Idnani, A. (1983): A numerically stable dual method for solving strictly convex quadratic programs, Mathematical Programming, Vol.27, 1-33.

Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H. (1983a): User's guide for SOL/NPSOL: a FORTRAN package for nonlinear programming, Report SOL 83-12, Department of Operations Research, Stanford University, USA.

Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H. (1983b): User's guide for SOL/QPSOL: a FORTRAN package for quadratic programming, Report SOL 83-7, Department of Operations Research, Stanford University, USA.

Han, S.-P. (1976): Superlinearly convergent variable metric algorithms for general nonlinear programming problems, Mathematical Programming, Vol.11, 263-282.

Han, S.-P. (1977): A globally convergent method for nonlinear programming, Journal of Optimization Theory and Applications, Vol.22, 297-309.

Hock, W., Schittkowski, K. (1981): Test examples for nonlinear programming codes, Lecture Notes in Economics and Mathematical Systems, Vol.187, Springer, Berlin, Heidelberg, New York.

Hock, W., Schittkowski, K. (1983): A comparative performance evaluation of 27 nonlinear programming codes, Computing, Vol.30, 335-358.

Lemarechal, C., Strodiot, J.-J., Bihain, A. (1981): On a bundle algorithm for nonsmooth optimization, in: Mangasarian, O.L., Meyer, R.R., Robinson, S.M.: Nonlinear programming, Academic Press.

Lindstroem, P. (1983): Algorithms for nonlinear least squares particularly problems with constraints, Report UMINF -

106.83, Institute of Information Processing, University of Umea, Sweden.

Powell, M.J.D. (1978a): A fast algorithm for nonlinearly constrained optimization calculations, in: Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Vol.630, Springer, Berlin, Heidelberg, New York.

Powell, M.J.D. (1978b): The convergence of variable metric methods for nonlinearly constrained optimization calculations, in: Nonlinear Programming 3, O.L. Mangasarian, R.R Meyer, S.M. Robinson eds., Academic Press, New York, San Francisco, London.

Powell, M.J.D. (1983): On the quadratic programming algorithm of Goldfarb and Idnani, Report DAMTP 1983/Na 19, University of Cambridge, Cambridge, Great Britain.

Schittkowski, K. (1980): Nonlinear programming codes, Lecture Notes in Economics and Mathematical Systems, Vol.183, Springer, Berlin, Heidelberg, New York.

Schittkowski, K. (1983): On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function, Mathematische Operationsforschung und Statistik, Series Optimization, Vol.14, 197-216.

Schittkowski, K. (1985): Solving constrained nonlinear least squares problems by a general purpose SQP-method, Report, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1985/86): NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems, Annals of Operations Research, Vol.5; 485-500.

Schittkowski, K. (1986a): ELL: A FORTRAN implementation of an ellipsoid algorithm for nonlinear programming, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1986b): MCO: A FORTRAN implementation of an interactive multicriterion optimization method, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1986c): DFNLP: A FORTRAN implementation of an SQP-algorithm for constrained nonlinear data fitting and min-max problems, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1986d): DFELL: A FORTRAN implementation of an ellipsoid algorithm for nonlinear data fitting, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1986e): QL: A FORTRAN implementation of a dual algorithm for quadratic programming, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1986f): LP: A FORTRAN implementation of the simplex algorithm for linear programming, User's Guide, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Schittkowski, K. (1987): Die Systementwicklungssprache SUSY, Report, Mathematisches Institut, Universitaet Bayreuth, 8580 Bayreuth.

Shor, N.Z. (1977): Cut-off method with space extension in convex programming problems, Cybernetics, Vol.12, 94-96.

Wilson, R.B. (1963): A simplicial algorithm for concave programming, Ph.D. Thesis, Graduate School of Business Administration, Harward University, Boston, USA.

EFFECTIVE ALGORITM FOR GLOBAL OPTIMIZATION WITH PARALLEL COMPUTATIONS

Roman G. Strongin, Yaroslav D. Sergeev
Gorky State University
Gorky, USSR

1. INTRODUCTION

Appearance of multiprocessor computer systems and local computer networks gives wide latitude for constructing optimization techniques using parallel iterations including simultaneous (due to many processors) computations (trials) of function to be optimized values at several points in a parameter space. Each trial appearing in sach parallel iteration could be performed at a separate processing unit using the same (shared or copied) program.

Let time-consumption of a single trial be substantially greater than time nedeed to select (following a decision rule) a group of points for next parallel iteration. Then it is also possible to neglect transfer times for exchanges between tigtly coupled processors (transfers of points of trials and reception of function values computed in these points). In this case the effectiveness of the parallel algorithm could be estimated by total amount of trials nedeed for solving a problem and by time period nedeed for this solution process (i.e. estimation by two criteria).

For a net formed by points of trials in a parameter space generated by a parallel algorithm it is naturally desirable to possess the same (or nearly the same) density as in the case of effective sequential algorithms (i.e. introduction of parallelism should not cause an appearance of redundant trials). With account of this condition it is desirable to use the largest possible number of processors to speed the searching for function optimum.

Following the above ideas we suggest some new parallel algorithms for global optimization. These methods are generalizations of effective sequential schemes from Strongin (1978) and Strongin and Markin (1986).

2. PARALLEL ALGORITHM FOR ONE-DIMENSIONAL UNCONSTRAINED PROBLEMS

The suggested algorithm for global minimization of a function $f(x)$, $x \in [a,b]$, based on the assumption that at any $l+1$ - st iteration (step) could simaltaneously be computed $p = p(l)$ function values at the points

$$x^{k+1}, \ldots, x^{k+p},$$

where $k = k(l)$ is a total number of function values computed in the course of l previous steps, i.e.

$k(1) = p(1) + \ldots + p(1)$.

Trials of the first step are carried out at any arbitrary $p = p(1) \geq 1$ internal points $x^1, \ldots, x^p$ of the given interval (a,b). The choice of points for any subsequent iteration is determined by the following rules:

(a) points $x^1, \ldots, x^k$ from previous trials are renumbered using subscripts in the following way:

$$a = x_0 < x_1 < \ldots < x_i < \ldots < x_k < x_{k+1} = b \qquad (1)$$

and associated with values $z_i = f(x_i)$, $1 \leq i \leq k$, computed at these points. The ordered sequence (1) is complemented by auxiliary points $x_0 = a$ and $x_{k+1} = b$ (values z_0 and z_{k+1} are undefined);

(b) the following value is calculated:

$$M = \max\left\{ /z_i - z_{i-1}/(x_i - x_{i-1})^{-1/n} : 1 < i \leq k \right\}. \qquad (2)$$

If (2) is undefined or equal to zero, it is assumed that $M = 1$;

(c) for each interval (x_{i-1}, x_i), $1 \leq i \leq k+1$, the value $R(i)$ (called <u>characteristic</u>) is computed, where

$$R(i) = (x_i - x_{i-1}) + (z_i - z_{i-1})^2/M^2(x_i - x_{i-1}) - 2(z_i + z_{i-1})/rM, \quad 1 < i \leq k, \qquad (3)$$

$$R(1) = 2(x_1 - x_0) - 4z_1/rM, \qquad (4)$$

$$R(k+1) = 2(x_{k+1} - x_k) - 4z_k/rM \qquad (5)$$

(here r is a parameter, with a value greater than 1);

(d) all characteristics are sequenced in the order of decrease:

$$R(i_1) \geq R(i_2) \geq \ldots \geq R(i_j) \geq \ldots \geq R(i_{k+1}); \qquad (6)$$

(e) next p trials of the $l+1$-st step are carried out at the points $x^{k+1}, \ldots, x^{k+p}$, where $x^{k+j} = S(i_j)$ and i_j, $1 \leq j \leq p$, are indices from the sequence (6),

$$S(i) = (x_i + x_{i-1})/2 - \operatorname{sign}(z_i - z_{i-1})(/z_i - z_{i-1}/M^{-1})^n/2r, \qquad (7)$$

$1 < i \leq k$. If $i = 1$ or $i = k+1$ the second term in the above formula is omitted.

It is supposed that

$$p = p(l) \leq k(l) + 1 \qquad (8)$$

for any value of $l \geq 1$. The <u>stopping</u> rule abrupts the

sequence $\{x^k\}$ if the condition $x_t - x_{t-1} \leq \beta$ is met, where $t = i_1$ from (7) and $\beta \geq 0$ is a given tolerance.

3. CONDITIONS FOR NON REDUNDANT SEARCH

It should be noted that the decision rules of the above algorithm are exactly the rules of the sequential method from Strongin (1978), if $p(1) = 1$, $l \geq 1$. Next theorem states some features of comparative efficiency of these two schemes (parallel and sequential) for the case, when in (2) and (7) $n = 1$ (this value have to be used if the function to be minimized is Lipschitz function).

THEOREM.

Let $\{x^k\}$ and $\{y^m\}$ be sequences of trial points generated correspondingly by parallel algorithm with coupled trials (i.e. $p(1) = 1$ and $p(1) = 2$, $l > 1$) and sequential algorithm while minimizing Lipschitz function with constant K (and tolerance in the stopping rule $\beta = 0$ - infinite search). Then:

(1) $\{x^k\} = \{y^m\}$ if $r \geq 2 + \sqrt{5}$ and constant $M \geq K$ is plugged into the formulas (3)-(5),(7) of the decision rule;

(2) $\{x^k\} \cap W = \{y^m\} \cap W$ if $r \geq 2$ and $f(x) = K/x - x^*/$, $x \in W \subset (a,b)$, where $x^* \in W$ is a point of absolute minimum of $f(x)$ in $[a,b]$.

Thus information about the results of previous iterations allows to determine not less then two points of next trials with the same efficiency (no redundancy) as in the choice of single trial in purely sequential algorithm.

4. GENERALIZATION FOR MULTIDIMENSIONAL CASE

The nested optimization scheme:

$$\min_{a_1 \leq x_1 \leq b_1} \cdots \min_{a_N \leq x_N \leq b_N} f(x_1, \ldots, x_N) \qquad (9)$$

allows to apply the suggested parallel algorithm for multidimensional optimization in a hyperinterval (by one dimensional optimization of nested one dimensional subproblems).

As follows from the theorem the above algorithm with coupled trials along with nested scheme (9) allows to use 2^N processors without causing redundancy (here N is a dimensionality of an optimization problem). In this case every iteration for any external problem (i.e. minimization in $[a_1,b_1]$) originates two nested (internal) problems. Each of these nested problems could also be optimized (in $[a_2,b_2]$) by parallel algorithm and so on.

To illustrate the possibilities of the proposed technique Table 1 presents some results of computer simulations of the search processes for two two-dimensional (i.e. $N = 2$) test functions with various number of processors (from 1 up to 9). In these experiments parameter $r = 1.7$ and relative tolerance in the stopping rule $\beta = 0.01$. Analizing the Table note that condition (8) causing equality $p(1) = 1$ could force some processors standing idle in partiqular iterations.

TABLE 1 Results of computer simulation

Number of processors	Parallelism External	Parallelism Internal	Number of steps (external)	Total number of trials	Time	Speeding	Redundancy
			1-st test function				
1	1	1	21	386	386	1	1
2	1	2	21	449	235	1.64	1.16
2	2	1	22	407	221	1.75	1.05
4	2	2	21	459	127	3.04	1.19
3	1	3	21	492	185	2.09	1.27
3	3	1	25	449	176	2.19	1.16
6	2	3	23	540	106	3.64	1.4
6	3	2	22	460	93	4.15	1.19
9	3	3	15	357	54	7.15	0.92
			2-nd test function				
1	1	1	25	526	526	1	1
2	1	2	25	541	283	1.86	1.03
2	2	1	17	373	210	2.5	0.71
4	2	2	17	379	108	4.87	0.72
3	1	3	25	573	216	2.44	1.09
3	3	1	21	474	197	2.67	0.9
6	2	3	17	411	85	6.19	0.78
6	3	2	19	441	94	5.6	0.84
9	3	3	20	498	74	7.11	0.95

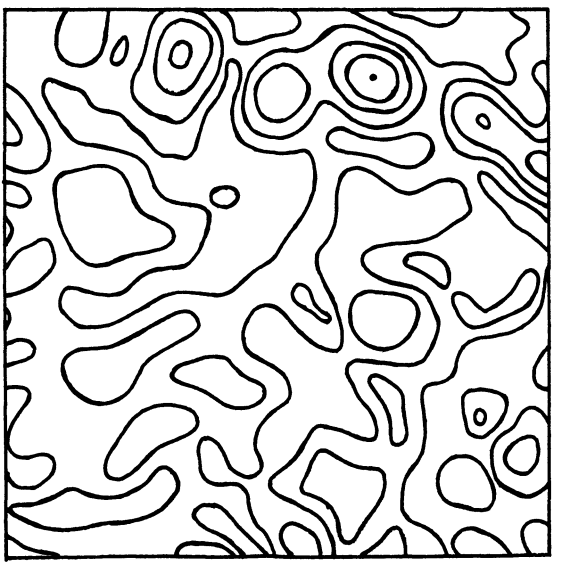


FIGURE 1 Level sets for the 1-st test function

Test functions used are essentially multiextrimal. Level sets for one of these functions are presented in Figure 1.

Another possible approach to multidimensional problems based on application of contineous space-filling curves $y(x)$ mapping a unit interval $[0,1]$ on the x axis onto N-dimensional hyper interval D. These mappings (called Peano curves) provide the origin for the scheme (Strongin, 1978):

$$\min\{ f(y): y \in D \} = \min\{ f(y(x)): x \in [0,1] \}.$$

Right-hand side of this expression could be computed by employing the considered parallel algorithm with $n = N$.

Similar parallel generalizations are possible for multidimensional multiextremal constrained problems by extending sequential algorithms from Strongin (1985) and Strongin and Markin (1986).

REFERENCES

Strongin, R.G. (1978) Numerical Methods in Multiextremal Problems. Nauka, Moscow (in Russian)

Strongin, R.G. (1985) Numerical Methods for Multiextremal Nonlinear Programming Problems with Nonconvex Constraints. In Lecture Notes in Economics and Mathematical Systems, vol.255, pp.278-282. Proceedings 1984. Edited by V.F.Demyanov and D.Pallaschke. Springer-Verlag. Berlin Heidelberg

Strongin, R.G., Markin, D.L. (1986) Minimization of Multiextremal Functions with Nonconvex Constraints. Kibernetika, (4): 64-69 (in Russian)

III. Graph Theory and Scheduling

M-MACHINE UNIT TIME SCHEDULING: A REPORT ON ONGOING RESEARCH

M. Bartusch*, R.H. Möhring, F.J. Radermacher*****

ABSTRACT:

Scheduling of partially ordered unit time jobs on m machines, aiming at minimal schedule length, is known to be one of the notorious combinatorial optimization problems, for which the complexity status is still unresolved. Availabe results give polynomial time algorithms for special classes of partial orders and for the two machine case. The present paper includes a general representation theorem on the existence of schedules of a given schedule length and a corresponding algorithmic method that allows a unifying approach to some of the solved special cases. It includes also a general algorithmic scheme that may work in the 3-machine case and may eventually lead to a polynomial time algorithm (or at least to membership in CO-NP) for the general m-machine case. Finally, it gives some insight into so-called m-critical posets, which should play an essential role in a proof on the correctness of the intended polynomial time algorithm.

KEY WORDS:

interval orders, m-critical posets, m-machine scheduling, makespan criterion, NP-completeness, polynomial time algorithm, three machine scheduling, trees, unit time scheduling.

---

* M. Bartusch († 1986), Lehrstuhl für Informatik und Operations Research, Universität Passau

** R.H. Möhring, Fachbereich Mathematik, TU Berlin

*** F.J. Radermacher, Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW), Ulm

1. INTRODUCTION

Scheduling of partially ordered jobs on m machines usually leads to **hard** (NP-complete) combinatorial optimization problems (cf. [17], [18] for an overview). This is true e.g. for **minimizing the makespan** (the schedule length) on two machines in the **nonpreemptive** case (even if only two types of job-durations are allowed [18]) and for minimizing the makespan on an arbitrary number of machines while all job durations are equal (**unit time jobs**) [18], [28]. Note that polynomial time solutions for the unit time case with integer starting points only correspond to polynomial solutions for the **preemptive** scheduling of jobs with arbitrary durations, as long as job durations are bounded by a polynomial in the number of jobs. This adds to the importance of the result that unit time jobs on two machines can indeed be scheduled in polynomial time, even if arbitrary release and due dates are involved [6], [9]. Polynomial time algorithms also exist for special 3-machine problems [9], they exist, furthermore, for any fixed number m of machines if the underlying poset is of bounded height [3] or of bounded width [1], [23], and it holds for arbitrary m, if the poset is a tree (forest) [14], an opposing forest [4] or an interval order [22]. However, inspite of intensive research by several scientists, the complexity status of the general m-machine unit time scheduling problem for $m > 2$, m fixed, still remains unresolved. In fact, it is one of the **notorious** problems in the field, mentioned already in the original list of ten basic open problems in complexity theory [10].

The present paper is a report on ongoing research concerning the general m machine problem. This research has been carried out by the authors now since about 5 years (cf. also [1], [23], [26]) and is guided by the conjecture that there should be a polynomial time algorithm for the general case with m being part of the exponent of the bounding polynomial. Within the paper, some normalizations, theorems and algorithmic tools are presented and discussed in connection with illustrative examples, which might be of particular use for other scientists working on the subject. The organization of the paper is as follows:

In Section 2, we start from a natural normalization of the problem (while covering some further generalizations such as profiles of available machines [3]) and then prove a quite general **representation theorem** (cf. [1]) that allows a decision on the existence of a feasible schedule of a given scheduling length by generalizing a well known **bounding approach**, known from the solution of the two machine case [8]. The representation leads to an algorithmic procedure for solving the general m-machine problem in $O(n^m)$ time depending, however, for $m > 2$, on the previous solution of a certain **choice** problem for which presently no general polynomial time solution is available. The choice problem is trivial for the 2-machine case and possible in an obvious way for some of the well-solved classes of posets such as interval orders and forests. The given algorithmic approach therefore makes it transparent, why the general case for $m > 2$ is so much harder to deal with then with the case $m = 2$ or certain special classes of partial orders.

The application of the presented algorithmic approach to some special cases of the problem under consideration is the subject of Section 3. We obtain results with the same performance bounds as the known solutions for the case m = 2 (even if release dates and due dates are involved), for special 3-machine problems and for the case of interval orders, where m is arbitrary. For forests and m arbitrary, we obtain a valid modification of the known solution, but while this has an almost linear performance bound, here the degree of the bounding polynomial is m.

In Section 4, we try to get around the **choice** problem in the general form, as it occurs in connection with the representation theorem. What we hope for is the identification of a sufficient core of information w.r.t. the placing of jobs, deducible in polynomial time, such that only performing those choices needed in the representation theorem which can be settled on the basis of this information is sufficient in the following sense: there is a feasible schedule for a given problem exactly if the bounds, computed by using the settled choices only, do not lead to a contradiction. We then show that if this conjecture is true, there is a polynomial time algorithm for the m-machine problem. Also we use as an example a reasonable core of information that essentially is derived via a **closure operator** and involves upper and lower bounds on the starting and completion times of sets of jobs and can be obtained in polynomial time. We use this to study and solve quite involved examples in order to demonstrate the strength of the approach as well as some interesting phenomena in these examples. Actually, this closure operator seems to be sufficient for the 3-machine case and should be of basic use also for the m-machine case in general. Possible extensions of the closure operator are also discussed.

Finally, in Section 5, we concentrate on approaches to prove the intended results w.r.t. the bounding techniques, discussed in Section 4. In doing so, we show that the proof can essentially be restricted to so-called **m-criticial posets** [26], i.e. posets that can optimally be scheduled on m machines in t time units, but taking away whatever immediate ordering relation from the poset allows scheduling it in t - 1 time units. In a certain sense, such posets are the hardest cases to deal with. Also, fast identification of (maximal) subposets of this type would show the problem to be in CO-NP. Actually, there is some hope that such posets may be so special in there structure that a proof becomes much easier. In fact this idea turned out to be correct in the 2-machine case, for which the 2-critical posets were very special indeed [26]. Their characterization in particular yielded annother way to obtain known polynomial time algorithms for the 2-machine case and, furthermore, also made the much harder situation for m > 2 better apparent. In the present paper now, we give another proof for the characterization of the 2-critical posets, then add to the information given in [26] some more examples and insights concerning 3-critical posets and, finally, give some evidence, that the complete characterization of the 3-critical posets might be possible and also might settle the matter for the 3-machine case completely.

2. A BASIC REPRESENTATION THEOREM

From a complexity theory point of view, the m-machine unit time scheduling problem is as follows:

Given a partial order $\theta = (A,0)$, where A is the set of jobs and 0 the ordering relation on A, and given some natural number $t > 0$, we ask whether there is a schedule for θ completed by time t, i.e. whether there is a mapping $T: A \to \mathbb{N}$, with $T(\alpha)$ denoting the starting time of job α, $\alpha \in A$, fulfilling:

(i) $0 \leq T(\alpha) \leq t - 1$ for all $\alpha \in A$,

(ii) $(\alpha,\beta) \in 0$ and $\alpha \neq \beta$ implies $T(\alpha) < T(\beta)$ and

(iii) $|\{\alpha \in A \mid T(\alpha) = i\}| \leq m$

for all $0 \leq i \leq t - 1$.

I.e. we ask, whether all jobs can be scheduled in such a way that ordering relations and machine availability are respected and all jobs are finished by time t. Of course, $|A| \leq m \cdot t$ is a necessary condition in order to be able to achieve this aim. In fact, one can easily normalize the problem to the case where $|A| = m \cdot t$, which essentially leads to the question of **whether it is possible to utilize all machines uninterruptedy up to time t** .

NORMALIZATION:

The m-machine unit time scheduling problem [(A,0), t] is polynomially equivalent to the special case where the number of jobs is exactly the number of available time-machine slots, i.e. to the case $|A| = m \cdot t$. The proof of the normalization is immediate by adding $m \cdot t - |A|$ independent (i.e. pair-wise incomparable) jobs to $\theta = (A,0)$ while leaving t unchanged. These independent jobs can be filled in the free slots remaining open after a solution of the original problem. The transformation is of course polynomial (even **linear**) in the original input length. Note however that the reduction may not be valid for special classes of posets, cf. also the remarks given in Section 3. E.g. for trees, it would lead to forests in any non trivial case, while for forests, in turn, the transformation remains in the class.

In the literature one if often concerned with more general problem formulations. For instance, there might be given **release dates** $r(\alpha) \in \mathbb{N}$ or **due date** $d(\alpha) \leq t$, for which any schedule must fulfill (iv) $0 \leq r(\alpha) \leq T(\alpha) \leq d(\alpha) - 1$ for any $\alpha \in A$, cf. [9]. Also, instead of m machines

being available over time there might be specified a so-called **m-profil** p, which gives for all time intervals [i, i+1), i = 0,1,2,...,t - 1 the number $p(i) \leq m$ of available machines and which corresponds to a more restrictive version of the contraints (iii), viz. (v): $|\{\alpha \in A \mid T(\alpha) = i\}| \leq p(i)$ for all i.

It is interesting to note that a **polynomial time algorithm** for the given normalized version of the m-machine unit time scheduling problem (m arbitrary, but fixed) yields, in fact, a polynomial time solution to the very general problem formulation (i), (ii), (iii), (iv), just given. This is obtained by the following.

GENERAL NORMALIZATION PROCEDURE:

Let a general m-machine unit time scheduling problem be given by a partial order (A,0), a time limit t, release dates r and due dates d and, finally, an m-profil p. Let $k := \sum_{i=0}^{t-1} p(i)$ and assume $|A| \leq k$. Construct the (m+1)-machine problem [(A´,0´), t], where $|A'| = (m + 1) \cdot t$ and where (A´,0´) is obtained from (A,0) as follows:

1. add an independent chain of new elements $\beta_1, \beta_2, \ldots, \beta_t$;
2. add k - |A| independent elements;
3. add for each time i = 0,1,...,t - 1 m - p(i) new jobs that are independent from all original jobs and the jobs introduced in (2), and that have release dates i and due dates i + 1, respectively.
4. add a new ordering relation (β_s, α) whenever a job α (including those introduced in (3)), has a release date $i \geq s$ and a new ordering relation (α, β_s) whenever a job α (including those introduced in (3)) has a due date $i < s$.

The idea behind the construction is as follows. Every schedule completing the modified problem in t time units on m + 1 machine will w.l.o.g. have to process the chain introduced in (1) on a particular machine, with job β_i being processed exactly in the time interval [i - 1, i). By the construction given in (3) and (4), exactly m - p(i) artificial jobs will also be executed in the time interval [i - 1, i). Thus exactly p(i) time slots will be left in this interval to schedule the original jobs. Again because of (4), original jobs can, because of the newly introduced ordering relations in (4), be only scheduled in accordance with their original release and due dates. Finally, the k - |A| independent jobs, introduced in (2) may fill the not used time slots in any solution that solves [(A, 0), t] for the given m-profil. Using these properties, it is easy to establish that any feasible schedule for the original problem extends to a solution for the normalization, while a restriction to the set A of any

feasible schedule for the normalization is a feasible schedule for the original problem. The transformation is obviously polynomial (even **linear**, except for the increase of the number of machines) and therefore settles the problem. Note, however, that for special classes of posets, the construction may again leave the special class.

The two normalizations discussed are a good motivation to study particularly the problem type $[(A,0),t]$ with $|A| = m \cdot t$ on m machines, as already proposed in [1]. The question for the existence of a feasible schedule is then equivalent to asking, whether all m machines can be utilized uninterruptedly in the time interval $[0,t]$. This formulation implies that if there is no such schedule, then - regardless of how we schedule - we will at some time $i = 0,1,2,...,t - 1$ end up with the following situation:

All machines were utilized uninterruptedly until time i while at time i, we can schedule at most $r \leq m - 1$ θ-independent jobs $\gamma_1,...,\gamma_r$ (where w.l.o.g. $m > 1$). Also, all remaining jobs are obviously successors of at least one γ_s, $s = 1,...,r$, for otherwise, an order-minimal job (i.e. a job without predecessors) among the remaining jobs could be scheduled at time i in addition to jobs $\gamma_1,...,\gamma_r$. Thus D, the set of the remaining jobs can be represented as $D = \bigcup_{j=1}^{r} N(\gamma_j)$, where $N(\alpha)$ denotes the set of successors of a job α. Note that it is impossible to schedule all jobs in D in the remaining time interval $[i + 1, t)$. This is evident because of $|D| / m > t - i$; in fact, we have $|D| = m \cdot (t - i) + (m - r)$. Note that we might determine in advance for any of the at most $O(n^m)$ many θ-independent sets B of jobs of cardinality $|B| = r < m$ the cardinality $|\bigcup_{\gamma \in B} N(\gamma)|$ of their successors and, in any case (or just in case $|\bigcup_{j \in B} N(\gamma)| = s \cdot m + (m-r)$ for some $s \in \mathbb{N}$) establish $t - (s + 1)$ as an upper bound $b^*(B)$ on the completion of the set B in a feasible schedule. Here, $b^*(B)$ being an upper bound on the completion of B is to be understood as the requirement that **at least one job from B** must be completed by time $b^*(B)$.

The given bound is imposed right away on a particular job α if $B = \{a\}$; we write $b^*(\alpha)$ in this case. As this is the only case occuring in 2-machine problems, this observation will later lead to a well-known polynomial time procedure for this case [8]. As soon as $m > 2$, for θ-independent sets B with two or more jobs, it will not be clear in general, which of the jobs in B should have to satisfy the obtained bound. If not for special reasons one of the jobs will have to be completed before the obtained bound anyway, we arrive at a **choice** problem, where the number of all combinations of choices will, in general, grow **exponentially** in the number of jobs. This is what makes the m-machine case for $m > 2$ so much more involved than the case of just 2 machines. The major conjecture in Section 4, however, will be that choices have to be taken into consideration

only in those cases in which only a particular job from the set B is able to accept the bound b*(B) in a feasible schedule, and that these cases can be identified in polynomial time. The following example will illustrate some of these aspects:

EXAMPLE 1:

Consider the following (3-critical; see below) poset $\theta = (A,0)$ with $A = \{1,2,...,12\}$, given in Figure 1 by a node-diagram, cf. also [26]. We ask, whether θ can be scheduled on 3 machines in 4 time units (which turns out to be impossible).

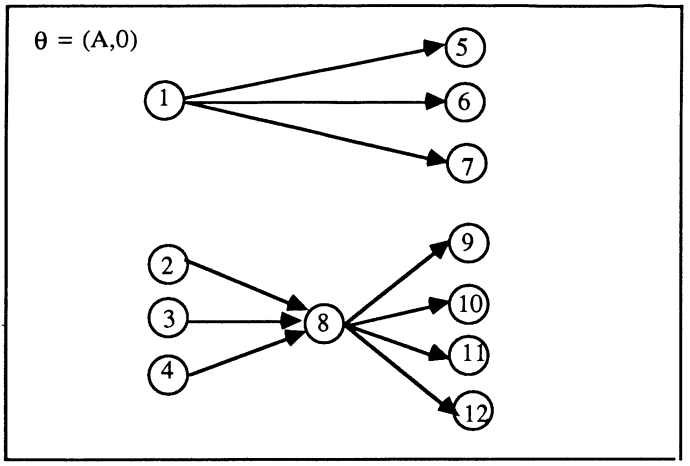


Fig. 1: A 3-critical poset

(1) Consider the partial schedule, where we start with jobs {1,2,3}, continue with {4,5,6} and then are stuck with jobs {7,8}. Here, job 8 has four successors, thus an individual bound $b^*(8) = 4 - 2 = 2$ and thus should have been completed earlier, i.e. by time 2.

(2) Consider therefore the alternative consisting in starting jobs {2,3,4} and then leading to the bottleneck {1,8}. Note that here both jobs have fulfilled there individual bound which is 3 for job 1 and 2 for job 8. Note also, that all three jobs 2, 3 and 4 have the individual bound 1. Now consider again the set {1, 8}. In total, both jobs have 7 successors, i.e. we have $b^*(\{1,8\}) = 4 - 3 = 1$. This means that one of the two jobs must be completed by time 1. This is impossible for job 8, as it has predecessors. Therefore, we obtain $b^*(1) = 1$. However, given that jobs 2, 3 and 4 have the same individual bound, we obtain a contradiction, i.e. θ cannot be scheduled on 3 machines in 4 time units.

The individual bound $b^*(\alpha)$, $\alpha \in A$, just introduced, can in fact be tightened in form of a bound b fulfilling $b(a) \leq b^*(\alpha)$ for all $\alpha \in A$, using a **backward induction**. Example 1 gives a motivation for that. Consider again job 2, which has 5 successors. This could be used to obtain the immediate bound $b^*(2) = 4 - 2 = 2$ for this job. However, with job 8 being a successor of job 2 and $b^*(2) = 2$, the sharper bound $b(2) = 1$ used in Example 1 (2) is obtained.

So, in general, we will define a sharper bound b **recursively** (cf. [8]), starting from maximal elements β of θ (for which $b(\beta) = t$), and then put for non-maximal elements α

$$b(\alpha) = \min_{i \in C} \{ i - \lceil |\{\beta \in A \mid \beta \in N(\alpha) \text{ and } b(\beta) \leq i\}| / m \rceil \},$$

where C is the set of all time points in $\{1,...,t\}$, such that there is a successor of α that has to be completed by that time. (Note that always $t \in C$). I.e. for all time points i, that are upper bounds for the completion time of certain successors of α, one has to take care, that the time interval $[b(\alpha), i)$ is long enough to handle all the respective jobs on m machines. The earlier bound $b^*(\alpha)$ is contained here in the minimization process for the case $i = t$.

It is immediate that no schedule can exist, as soon as for some time i, the number $|\{\alpha \in A \mid b(\alpha) \leq i\}|$ strictly exceeds $m \cdot i$; cf. also the argument used in Example 1(2), where $|\{\alpha \in A \mid b(\alpha) \leq 1\}| > 3 \cdot 1$ was fulfilled. In fact, the converse is also true for the 2-machine case, i.e. there is a feasible schedule, as soon as the condition $|\{\alpha \in A \mid b(\alpha) \leq i\} \leq m \cdot i$ is fulfilled for all $i = 0,1,...,t - 1$. This is actually the basis for known polynomial time algorithms for the 2-machine case [8] and will prove to be a special case of the **basic representation theorem**, following next. This representation theorem will extend the 2-machine case approach to the general m machine unit time scheduling problem. To achieve this, we will have to replace $b^*(B)$ by the sharper bound $b(\alpha)$, obtained by backward induction, also in case of $|B| > 1$. The main obstacle is that we do not immediately know, how to handle bounds of this type at all, i.e. what to do with them, how to adapt them and how to relate them to certain jobs in the respective sets B. Our approach will intuitively be to focus at the following aspect. Given any feasible schedule for the problem under consideration and given any correct bound deducible for a θ-independent set B with $1 \leq |B| < m$ that is consistent with the given schedule, there will be at least one element α in B that observes this bound. Now pick a particular such element α_B from each such set B and call the function $B \rightarrow \alpha_B$ (for all B considered) a **choice**. Note that ,intuitively speaking, α_B being the choice for B means that the bound for α_B, defined next, will have to take into account all sucessors of elements in B. In order to do this, the bounds must already been defined for **all** these successors. It may happen that, given a particular choice, some elements can **block** each other in the sense that there is a **circular dependency** among them concerning the need of knowing already their respective bounds (cf. a simular phenomena in connection with strategies, induced by so-called

selections [15], [16] in stochastic scheduling models). We call a choice **consistent** or **non-blocking** , if such a circular dependency does not exist. This is equivalent to the existence of a sequence among the jobs that allows a **backward recursion** as needed in the following definition of a **choice dependent bound for non-blocking choices** that is analogous to the individual bound discussed above:

(1) Put $b(\alpha) = t$ for all maximal elements α

(2) Let α be non-maximal; α is the chosen element for certain considered θ-independent sets B, in particular α is chosen for the set $B = \{\alpha\}$.
Put then $b(\alpha) := \min \{b(B,\alpha) \mid \alpha \text{ chosen for } B\}$ where
$b(B,\alpha) := \min_{i \in C(B)} \{i - \lceil |\{\beta \in \bigcup_{\gamma \in B} N(\gamma) \text{ and } b(\beta) \leq i\}| / m \rceil\}$.

Here, $C(B)$ is the set of all time points i in $\{1,...,t\}$ for which there is a successor of some $\gamma \in B$ having an individual time bound smaller or equal to i. Note that w.r.t. all $b(B,\alpha)$, α chosen for B, $\bigcup_{\gamma \in B} N(\gamma)$ will be non-empty, as there will be successors of α. Also, the bounds for all jobs in this set will already be determined because of the choice being non-blocking and the sequencing within the backward recursion was chosen accordingly, and, finally, all $\beta \in \bigcup_{\gamma \in B} N(\gamma)$ will certainly fulfill $b(\beta) \leq t$, i.e. $t \in C(B)$ will be valid for all considered sets B. Obviously, the given bound is sharper than the earlier bound computed on the basis of individual elements only and also $b(B,\alpha) \leq b^*(B)$ is obvious.

As was already mentioned above, an easy inductive argument shows the following: starting from a feasible schedule T, a **consistent choice** for all considered sets B can be made such that the given schedule observes the corresponding choice dependent bound, i.e. such that $T(\alpha_B) + 1 \leq b(\alpha_B) \leq b(\alpha_B, B)$ is fullfilled for any considered B and any chosen α_B. To achieve this, one merely has to chose for any B an element within this set with **earliest** starting time w.r.t. T. This consistency of T with the bounds b means, in particular, that w.r.t. these bounds, a contradiction of the type $|\{\alpha \in A \mid b(\alpha) \leq i\}| > m \cdot i$ for $i \in \{0,1,...,t-1\}$ can again not occur.

The following theorem now shows that again the converse is also true:

BASIC REPRESENTATION THEOREM

Let [(A,0), t] denote an m-machine unit time scheduling problem fulfilling $|A| = m \cdot t$. Then the following holds:

There is a feasible schedule for the problem, iff there is a **non-blocking choice** α_B for each θ-independent set B of jobs with $1 \leq |B| < m$, such that w.r.t. the correspondent bounds b, there is no contradiction of the form $|\{\alpha \in A \mid b(\alpha) \leq i\}| > m \cdot i$ for some $i \in \{0,1,...,t-1\}$.

If there is a feasible schedule, it may e.g. be constructed by successively starting in the i-th. time slot, $i = 1,2,...,t$ any set of m jobs among the respective set of minimal jobs remaining, such that w.r.t. the bounds $b_i(\alpha) := b(\alpha) + 1 - i$ on the remaining jobs, there is no bound contradiction as considered. Suitable sets of jobs will be available in any time slot. This means, in particular, that we can do **priority scheduling** w.r.t. the priority list induced by increasing values of b.

Proof:

The argument for the easy direction was already given above. So assume there is a **non-blocking choice** α_B for all θ-independent sets B with $1 \leq |B| < m$, such that there is no bound contradiction $|\{\alpha \in A \mid b(\beta) \leq i\}| > m \cdot i$ for $i \in \{0,1,...,t - 1\}$ w.r.t. the correspondent bounds $b(\alpha)$, $\alpha \in A$, defined above. We then have to show that there is indeed a feasible schedule for this problem. The proof is given by induction on t, where the case $t = 1$ is trivial. So assume the result to be true for a certain value of t and consider the case of time limit $t + 1$ and $m \cdot (t + 1)$ elements.

Assume first that the set M of minimal elements in θ has strictly less than m elements and let $\alpha_M \in M$ denote the chosen element within M. Obviously, we obtain $b(\alpha_M) \leq b(M, \alpha_M) \leq t - t = 0$, and thus $|\{\alpha \in A \mid b(\alpha) \leq 0\}| > 0$, a contradiction.

Consequently, we can find a set S of m elements to start with. Note that after removing any such m elements, we come to a subproblem with a base set A', where the question is to schedule the remaining $m \cdot t$ elements in t time units on m machines. For this subproblem, the induction hypothesis can be applied. We try this w.r.t. the restriction of the original choices made for jobs and sets of jobs within the subproblem. If we compute the bounds b' for the subproblem and take into account the backward recursion, it is easily seen that these are greater or equal to the original bounds b, reduced by 1, i.e. we have $b'(\alpha) \geq b(\alpha) - 1$ for all $\alpha \in A'$. Here, the reduction by one is due to fact that the recursion in the subcase starts from $b(\beta) = t$ instead from $b(\beta) = t + 1$ for maximal elements and is otherwise basicly identical, while a possible increase comes inductively from the elemination of those considered sets B that contain some of the unstarted elements but had, as chosen element α_B, an element belonging to A'. Assume now, there is no bound contradiction of the form $|\{\alpha \in A' \mid b'(\alpha) \leq i\}| > i \cdot m$ for $i \in \{0,1,...,t - 1\}$ for the subproblem. Then by induction, we can schedule the jobs in A' in t time units, leading to the desired schedule of all jobs in A in $t + 1$ time units. Thus we are done, whenever we can start with m elements that avoid a

bound contradiction for the remaining problem. We will show in the following that at least one such start set always exist, viz. a set S of jobs with the **lexicographically smallest** (sharpest) bounds. In fact, we will even show that taking away this set avoids even any contradiction for the subproblem w.r.t. the sharper bounds $b(\alpha) - 1$, $\alpha \in A'$. Thus the proof allows us to schedule jobs according to the **priority list**, induced by the originally computed bounds $b(\alpha)$, $\alpha \in A$, as stated in the theorem.

Assume that this is not true, i.e. there is a bound contradiction $|\{\alpha \in A' \mid b(\alpha) - 1 \leq i\}| > i \cdot m$ for some $i \in \{0,1,...,t - 1\}$. Certainly, then $\{\alpha \in A' \mid b(\alpha) - 1 \leq i\} \subseteq \{\alpha \in A \mid b(\alpha) \leq i + 1\}$, where $i + 1 \in \{0,1,...,t\}$. If we assume that $b(\beta) \leq i + 1$ is true for all jobs in the starting set, we obtain a bounding contradiction for the original problem. Therefore, there is a proper subset B with $|B| < m$ among the started jobs (i.e. $B \subset S$), which contains exactly those started elements β, fulfilling $b(\beta) \leq i + 1$. B is not empty, as an element with the smallest bound was started and necessarily belongs to B. Let α_B denote the chosen element in B. Then we obtain $b(\alpha_B) \leq b(B,\alpha_B) = (i + 1) - (i + 1) = 0$ because of the immediate fact that all elements from $\{\alpha \in A' \mid b(\alpha) - 1 \leq i\}$ must be **successors** of elements in B. This follows, as none of these element can be minimal in θ, due to the fact that jobs with greater bounds (even bounds greater than $i + 1$) belong to the lexicographically smallest starting set, viz. are the elements of the set $S \setminus B$. Furthermore, these jobs from $S \setminus B$ cannot be predecessors of jobs β fulfilling $b(\beta) \leq i + 1$ because of **monotonicity** reasons w.r.t. the bounds (i.e. by construction, the bound of a predecessor has always to be smaller than the bound of a successor). Therefore, we indeed obtain the inequality $b(\alpha_B) \leq 0$ which constitutes a bound contradiction, concluding the proof. ■

The proof shows that bounds $b(B)$ for θ-independent sets B of jobs with $1 \leq |B| < m$ must only be determined if a situation occurs in which, for a certain time bound i, all jobs fulfilling $b(\alpha) \leq i$ belong to the set $B \cup \bigcup_{\gamma \in B} N(\gamma)$ and, in addition, fulfill $|\bigcup_{\gamma \in B} N(\gamma)| / m > i - 1$. As a consequence of this observation, the consideration of choices for the computation of the respective bounds can be restricted to a subset of all sets B considered so far, whenever contradictions of the type considered can only happen for sets B within this subset; cf. in particular Lemma 4. I.e. the Representation Theorem remains valid for the bounds b computed on the basis of such a restricted set of considered sets B with $1 \leq |B| < m$. This is the statement of the following Corollary.

COROLLARY 2:

Let M denote a subset of the set of all θ-independent sets B of jobs with $1 \leq |B| < m$, such that contradictions of the form $[b(\alpha) \leq i$ for some $i = 1,...,t$ implies $\alpha \in B \cup \bigcup_{\gamma \in B} N(\gamma)$ and $|\bigcup_{\gamma \in B} N(\gamma)|/m > i - 1]$ can only occur for sets $B \in M$. Then the Represenation Theorem remains valid, if choices and bound computations are restricted to the consideration of sets in M only.

In the following, we briefly discuss the consequences of the Representation Theorem for an **algorithmic treatment** of the problem. In this context, we first observe that (w.r.t. the number n of jobs) $O(n^m)$ is about the best complexity bound one can probably hope for in the general m-machine case. For 2 machines, this is the exact bound in the known solution [8]. Furthermore, the 2-machine case with release and due dates, which can be regarded as a special version of the 3-machine case via the general normalization procedure given, requires an $O(n^3)$ time bound in the known solution [9]. Annother motivation for $O(n^m)$ as a lower bound on the needed computation time in the apparent need to consider all independent sets of jobs with cardinality up to $m - 1$, which are $O(n^{m-1})$ many, and the need to do some involved computations w.r.t. all these sets.

We will discuss now that in a certain sense then, a solution for an m-machine problem could essentially be interpreted as the task of finding a **test-choice** for such a problem, i.e. to determine a choice with the following property: either there is a schedule compatible with the choice, or there is no feasible schedule at all. If this can be achieved in $O(n^s)$ time with $s \geq m$ arbitrary, the general m-machine problem can be solved in $O(n^s)$ time. I.e. all the other steps arising from the Representation Theorem, such as **testing the consistency** of the given choice, **computation** of the associate **bounds, testing** the considered **bound condition** and, finally, producing a **feasible schedule** can be done in $O(n^m)$ time, $m \geq 1$ arbitrary. As an application, we will demonstrate in Section 3, how some of the solved special case of m-machine unit time scheduling can, in fact, be obtained via a solution of the test-choice problem, followed by an application of the Representation Theorem.

CHOICE VERIFICATION ALGORITHM

Given a m-machine unit time scheduling problem $[(A, 0), t]$ with $A = m \cdot t$, a set M of θ-independent sets of jobs B with $1 \leq |B| < m$ as required for consideration, and an arbitrary choice $B \rightarrow \alpha_B$ for all $B \in M$, it is possible to check in $O(n^m)$ time whether there exists a valid schedule compatible with this choice and to produce such a schedule, if it exists. In particular, checking includes the following steps:

1. Checking for $m > 2$ in $O(n^m)$ time, whether the given choice is **non-blocking**, and if it is not, determining a valid sequence for the backward recursion.

2. Computing the choice-dependent bound in $O(n^m)$ time, using the obtained sequence for the backward recursion, if the choice turned out to be non-blocking,

3. Checking in $O(n^2)$ time, whether there is a bound contradiction of the form $|\{\alpha \in A \mid b(\alpha) \leq i\}| > m \cdot i$ for some $i \in \{0,1,2,..., t - 1\}$,

4. Producing in $O(n^2)$ time a feasible schedule compatible with the given choice (in case there was no bound contradiction) by doing priority scheduling w.r.t. the obtained bounds b.

We shortly argue, how the described steps have to be organized in order to obtain the given time bounds. The validity of the whole procedure is of course due to the Representation Theorem.

We start from a choice being given in the form of a list $B \rightarrow \alpha_B$ for all $B \in M$, where $|M|$ is bounded by $O(n^{m-1})$. In $O(n^m)$ time, we can, using bucket sort, reorder this list into the form $\alpha_1, M_1, \alpha_2, M_2, ..., \alpha_{n-1}, M_{n-1}, \alpha_n, M_n$, where the sequence of jobs is arbitrary and where M_i gives the (possibly empty) sets in M, for which α_i is the choice. Note that the whole list has length $O(n^{m-1})$, while each $|M_i|$ is bounded by $O(n^{m-2})$.

Also, as part of the data structures involved, we keep a list of the present bound values $b(\alpha)$ for all $\alpha \in A$, with $b(\alpha) = t$ as common start value. We will also maintain a modified list in which time points $i = 1,2,3,4,...,t$ are followed by those jobs which currently have the bound i. Note, finally, that a major step will be the determination of the union of two sets of successors in a particular form (merging of sorted lists with preprocessing via bucket sorts). This **union operator** will need **linear time** and will be defined as follows: successor jobs are arranged in advance in increasing order concerning the given numbering of jobs. Jobs are picked up one by one (at most n many) from each of the two involved successor sets, their present bound is taken from the current bound list and the element is then put into a list (if not already there) of jobs in the union, ordered as follows: after each time point i those jobs with present bound i follow in increasing order.

Step 1:
Given the preparatory data organisation, we check for $m > 2$, whether the choice is non-blocking (blocking is impossible for $m \leq 2$). In order to do this, we determine the relation R on A defined by $(\alpha, \beta) \in R: \Leftrightarrow [\beta \in N(\gamma)$ for some $\gamma \in B, B \in M$, and α is chosen for B]. I.e. $(\alpha, \beta) \in R$ iff the bound for β must be known in order to compute the bound for α. R is obtained by applying

for all α and all sets $B \in M_\alpha$ the union operation, consuming linear time. I.e. for each α, at most m times $O(n^{m-2})$ applications will be performed. This leads to an $O(n^m)$ time complexity in order to determine R. It is easily verified that the considered choice is non-blocking iff R is **acyclic** . This property of R can be checked by standard methods in $O(n^2)$ time. If R is acyclic, then obviously, the dual of any **linear extension** of R is appropriate as a sequence for the backward recursion. Such a sequence can again be determined in $O(n^2)$ time by standard methods.

Step 2:

We determine the bounds b, starting from the original modified list of at most $O(n^{m-1})$ considered sets B. In doing so, we use the sequence among the jobs determined in Step 1 for the backward recursion. What we essentially have to achieve is to determine in **linear time** for any considered set B, α being the choice for B, the number

$$b(B, \alpha) := \min_{\gamma \in C(B)} \{ i - \lceil | \{ \beta \in \bigcup_{\gamma \in B} N(\gamma) \text{ and } b(\gamma) \leq i \} | / m \rceil \}.$$

It is easily seen that, once this is achieved, $b(\alpha)$ itself is obtained in $O(n^{m-1})$ time (remember $|M_\alpha| \leq O(n^{m-2})$) and thus b is achieved in $O(n^m)$ time, as stated.

Now it is easily seen, that the determination of $b(B,\alpha)$ involves m applications of the linear time union operator described above, where the operator itself makes use of the available list of all (required) present bounds. This list will be updated with $b(\alpha)$, once $b(\alpha)$ is determined. After having applied the union operator m times (m fixed), we go through the resulting list of jobs in linear time. We will see for each time point $i = 1,...,t$, in increasing order, whether there are jobs in $\bigcup_{\gamma \in B} N(\gamma)$ accepting this bound (while not accepting the previously considered smaller bounds bounds), we count the number of such jobs, add this number to the total number of jobs found already earlier for time bounds up to i - 1 (where a sum greater to zero exactly means $i \in C(B)$), compute the time bound associate with i, update the minimum over all such bounds up to this moment, and proceed to i + 1. This essentially concludes Step 2.

Step 3 and 4:

These two steps are not time-criticial and can be done in a straightforward way.

In summary, the Representation Theorem delivers an algorithmic tool which, in $O(n^m)$ time, allows us to check whether any particular **conflict solving mechanism** on sets of jobs that might eventually turn out to be bottlenecks within a schedule, really works. The use of such a test device for special problem instances as well as for the general case will be the subject of Sections 3 and 4.

3. SOME WELL-BEHAVED SPECIAL CASES

In this section, we give five applications of the Representation Theorem to well-behaved special cases of m-machine unit time scheduling problems. These cases are the 2-machine case (Lemma 3), a special 3-machine case (Lemma 4), a scheme for the general incorporation of release and due dates (Lemma 6), interval orders (Lemma 7) and forests (Lemma 8).

LEMMA 3: (cf. also [2], [6], [7], [8], [9])

2-machine unit time scheduling is possible in $O(n^2)$ time.

Proof:

As argued in Section 2, the Representation Theorem applies to the normalized version of the 2-machine case without any choice problem at all. Thus Steps 2. - 4. of the choice verification algorithm solve the problem in $O(n^2)$ time, i.e. either establish inconsistency of the problem or give a feasible schedule. The time bound is the best worst case bound known w.r.t. the number of jobs (cf. [6], [8]), in fact, the algorithm coincides with the algorithm given in [8]. For a somewhat improved algorithmic approach cf. [7]. ■

LEMMA 4: (cf. [9])

A 3-machine unit time scheduling problem [(A,0), t], with (A,0) containing a chain $\beta_1, \beta_2, \ldots, \beta_t$ of length t, can be solved in $O(n^3)$ time. [Note that via the general normalization procedure given, $|A| = m \cdot t$ can be assumed w.l.o.g. and also, any 2-machine unit time scheduling problem with release and due dates can be brought into this form, cf. also [9].

Proof:

The proof consists in three major steps: first we identify (implicitly) an appropriate system M of sets to be considered, then we give an associated test-choice and, finally, we argue how to modify the choice verification algorithm in order to cope with this implicit characterization of the system M (and the corresponding test choice) in a way that does not increase the time complexity $O(n^3)$.

Step 1: Definition of M :

We start from the observation that if the given problem has no solution then any possible fixing of M and any choice w.r.t. M will lead to a bound contradiction because of the Representation Theorem. If instead there is a solution, then w.r.t. any successful choice, job β_i from the given chain of length t is scheduled in the time interval [i - 1, i], i = 1,...,t. (Also, β_1 has no predecessors and β_t no successorss). These facts impose an a **priori bound** of $b(\gamma) := \min\{i \in \{1,...,t-1\} \mid \beta_{i+1} \in N(\gamma)\}$ on all jobs $\gamma \in A$ fulfilling $(\gamma,\beta_i) \in 0$ for some i = 2,...,t. The respective elements and bounds can be determined in $O(n^2)$-time beforehand. Also it can be checked whether $\{\alpha \in A \mid b(a) \le i\} \le 3 \cdot i$ is fulfilled for all i = 0,1,...,t - 1 w.r.t. the a priori bound. If not, the problem has no solution. If the condition is fulfilled, we ask which sets B of θ-independent jobs with $1 \le |B| \le 2$ have to belong to M in order to allow an application of the Representation Theorem in the sense of Corollary 2. The inductive argument from the proof of the Representation Theorem shows that exactly those θ-independent sets B with $1 \le |B| \le 2$ have to be considered for which, after having used all machines uninterruptedly in an appropriate way up to time j - 1, B becomes a bottleneck in the time interval [j - 1,j) in the sense that w.r.t. the intended bound b, the condition $|\{\delta \in \bigcup_{\gamma \in B} N(\gamma) \mid b(\delta) \le i\}| / 3 \le i - j$ for some i = j + 1,...,t is violated. We implicitely define M to be just the set of all those potential bottleneck sets B. Obviously, all these sets B are either of the form $\{\beta_j\}$ for j = 1,2,...,t - 1 or of the form $\{\beta_j,\alpha\}$, j = 1,...,t with α θ-independent of β_j and $b(\alpha) \ge j$. Unfortunately, we do not know in advance which of the given sets really belong to M. So we have an additional problem to cope with when discussing the algorithmic treatment in Step 3.

Step 2: The test choice for M

The choice for a set $\{\beta_j\} \in M$ is of course β_j, while for sets $\{\beta_j, \alpha\} \in M$ it is α. This is, in fact, the only potentially successful choice at all, as any other choice for a set $\{\beta_j,\alpha\} \in M$ would imply $b(\beta_j) \le j - 1$ and thus $b(\beta_1) \le 0$, which yields an immediate bound contradiction. (The given choice is non-blocking; the dual of any linear extension of (A,0) could be used for a backward induction).

Step 3:

A variant of the choice verification algorithm for the given, implicitly defined set system M.

(1) We start with the described a priori bound for jobs that are predecessor of some β_j, j = 2,...,t and with $b(\alpha) = t$ for all other jobs. For maximal elements of (A,0), the bound $b(\alpha) = t$ will never

change. For all other jobs, the bound will be updated from time to time, always keeping the least value obtained so far. The main idea is to scan the sets B of type $\{\beta_j\}$, $j = 1,...,t - 1$ and $\{\alpha, \beta_j\}$, $j = 1,...,t$, that may belong to M. The aim is to identify, whether such a set really belongs to M, i.e. whether it leads to a bottleneck of the form $|\{\delta \in \bigcup_{\gamma \in B} N(\gamma) \mid b(\delta) \leq i\}| /3 > i - j$. As soon as the bottleneck-situation is identified, the corresponding bound $i - \lceil |\{\delta \in \bigcup_{\gamma \in B} N(\gamma) \mid b(\delta) \leq i\}| /3 \rceil$ is imposed on α in the updating process.

(2) The main problem in (1) is how to cope with the problem of having to rely on knowledge about $b(\delta) \leq i$, which itself depends on the previous identification of (parts of) M. A solution to this difficulty is obtained by the following observation. In order for $B = \{\alpha,\beta_j\}$ to be a member of M, a contradiction of the form $|\{\delta \in \bigcup_{\gamma \in B} N(\gamma) \mid b(\delta) \leq i\}| /m > j - i$ for $i \in \{j + 1,...,t\}$ is needed, where, in addition $j \leq b(\alpha) < i$. Here, $b(\alpha) < i$ arises from the fact that if no successor δ of α fulfills $b(\delta) \leq i$, $b(\beta_j) \leq j - 1$ would follow, leading to the bound contradiction $b(\beta_1) \leq 0$. Certainly, if we would know in the above situation the bounds for all successors of α and β_j, we could check, whether there is a bound contradiction for M. If it occurs, this would result in an updating of $b(\alpha)$, leading to a value $b(\alpha) \leq j - 1$.

The intended procedure will make use of this observation and needs the notion of the **dual height** $h(\alpha)$ of a job α w.r.t. (A,0), where $h(\alpha)$ denotes the cardinality of a maximal chain in (A,0), starting from α. The dual height of all jobs can be computed beforehand by standard methods in $O(n^2)$ time. We can check within the same time bound whether the dual height of each β_i is $t - i + 1$, $i = 1,...,t$. If this is not the case, there is no solution. If instead it is true, we know that the dual height of all successors of β_i must be less or equal to $t - i$.

Notice now that for all elements of dual height 1, i.e. for all maximal elements δ, the bound $b(\delta) = t$ is correct and will never change. This allows us to check all sets $\{\beta_t,\alpha\}$ for all $\alpha \in A$ with $h(\alpha) = 2$. In all cases this will lead to a new bound $b(\alpha) < t$. We next check $B = \{\beta_{t-1}\}$. If this yields the necessity to decrease $b(\beta_{t-1})$, the problem has no solution. Otherwise, we consider sets $\{\beta_{t-1},\alpha\}$ with $h(\alpha) = 2$ in those cases, where $b(\alpha) = t - 1$. This may lead to a smaller bound $b(\alpha) < t - 1$. We proceed with sets $\{\beta_j,\alpha_3\}$, $i = t - 2, t - 3,...,1$ and jobs α of dual height 2 with present bound $b(\alpha) \geq i$, always checking before the set $B = \{\beta_i\}$ w.r.t. inner time bound t, to see, whether there is no contradiction in order to proceed.The bound computation w.r.t. $\{\beta_i,\alpha\}$ is also restricted to the **inner bound time t** (i.e. to the case $i = t$ in the given formula) and needs **linear time**, cf. the choice verification algorithm, i.e. we only need to see whether $|N(\alpha) \cup N(\beta_i)| /3 > t - i$ in order to decide, whether $\{\beta_i,\alpha\} \in M$. Computing, if required, the associated bounds, we finally arrive at the correct bound $b(\alpha)$ for α, which in turn, will be used to update the bound of predecessors of α by comparision with $b(\alpha) - 1$. I.e. to summarize, we can compute the correct bound for all jobs of dual height 2 because w.r.t. their successors δ, which are all maximal, the

bounds $b(\delta) = t$ will not change. Consequently, computations are restricted to inner bounds t only. Note that after this step, the bounds for all jobs of dual height smaller or equal to 2 will be known. Also note, that sets $\{\beta_i,\alpha\}$ were considered **at most once**.

If we proceed now to jobs of dual height 3, there will (after a positive check for $\{\beta_{t-2}\}$ and inner time points $i = t - 1, t$) occur the additional difficulty that such a job α may have a successor with a bound $b(\alpha) = i$, $i \leq t - 2$, i.e. a current bound of $b(\alpha) \leq i - 1 < t - 2$. If we now would try to consider $B = \{\beta_{i-1},\alpha\}$, the needed bounds of certain successors of β_{i-1} might still be missing. We therefore will treat only those jobs of dual height 3, whose present bound b is equal to $t - 2$ and postpone jobs with present bound $t - i$, $i > 2$, until the consideration of jobs of dual height $i + 1 > 3$; note that the bound of jobs of dual height 3 is necessarily less or equal to $t - 2$. This procedure then leads to the following **general step**.

Assume we have scanned iteratively all jobs of dual height 1,2,3,...,r. In doing so, a job α from this set either obtained his correct bound $b(\alpha) \geq t - r + 1$, or it has a current bound $b(\alpha) \leq t - r$. We then go over to $r + 1$. This means that we first check for the set $B = \{\beta_{t-r}\}$, whether it necessitates a decrease of the bound of β_{t-r}, which means that there is no solution to the problem. Note that all successors of β_{t-r} were already scanned. If any of these successors δ has a present bound $b(\delta) \leq t - r$, the problem would have again no solution. Otherwise, the bounds of all successors are correct and the required computations can be done for all inner time bounds $i = t - r + 1, t - r + 2,...,t$ simultaneously in **linear** time (cf. the choice verification algorithm). Again, if the bound of β_{t-r} should be decreased, the problem has no solution. Otherwise, the bound $b(\beta_{t-r}) = t - r$ and the bounds $b(\delta) \geq t - r + 1$ of all successors of β_{t-r} are correct.

In this case, we now consider for $j = t - r, t - r - 1,...,2$ sets $B = \{\alpha,\beta_j\}$ for all those $\alpha \in A$, for which the previous bound $b(\alpha)$ equals $t - r$. In doing so, we consider (only) the inner time points $t = t - r + 1,...,t$, because α has no successors with a bound smaller or equal to $t - r$. We scan those jobs in a sequence of increasing dual height, where the dual heights involved maybe $2,3,...,r + 1$. Note that once a (first) set $\{\beta_j,\alpha\} \in M$ is identified, it leads to a present bound $b(\alpha) \leq t - j - 1$ for α and $b(\delta) \leq t - j - 2$ for all predecessors of α. (Thus, in particular, all elements δ of dual height $r + 1$ have a bound $b(\delta) \leq t - r$). If the bound of α was decreased, we postpone the further treatment of α until we get to treatment of the respective greater dual height). To see the correctness of the procedure one only has to observe, that if a set $B = \{\alpha,\beta_j\}$ with $j \leq t - r$ is considered, we need only take into account inner time bounds $i = t - r + 1, t - r + 2,...,t$ (as α has no successor δ fulfilling $b(\delta) \leq t - r$). Now while the bounds for elements in $N(\alpha)$ are already known, we know that all those elements δ in $N(\beta_j)$ whose bound is not known certainly fulfill $b(\delta) \leq t - r + 1$. This is obviously enough to carry out the required computations. Again, as demonstrated in connection with the choice verification algorithm, all computations needed in connection with B can be performed in **linear** time. If, in this process, no bound contradiction is

obtained, the correct bound $b(\alpha) = t - r$ of α is identified, i.e. this value can never change again. Otherwise, we obtain for some first $j \leq t - r$ a bound $b(\alpha) \leq t - r - 1$ and will consider the job again, later, as described above. Note that here again, sets $\{\beta_j,\alpha\}$ are only considered **once**. For, either there will be no bound decrease for α for all $j \leq t - r$, in which case the correct bound of α was obtained, or else, there is a first j that leads to a bound decrease. In this case, we will later on continue with sets $\{\beta_j,\alpha\}$ where $j < j'$. This concludes the modified choice verification algorithm.

(3) W.r.t. the needed computation time, one realizes that in a certain sense the given procedure is a kind of reordering of the original choice-verification algorithm. The potentially $O(n^2)$ many elements of M are looked through in a different order, not partitioned into blocks depending on the chosen α. The modification was however done in a way that still achieves the time complexity $O(n^3)$. The main feature to this aim is the already demonstrated fact, that each of the n sets $\{\beta_j\}$ is considered once, and that similarly each of the at most $O(n^2)$ many considered sets $\{\beta_j,\alpha\}$ is taken into account **at most once** , where the computational effort in connection with each considered set is **linear**. This yields the stated complexity bound $O(n^3)$. ■

EXAMPLE 5:

Consider the 3-machine unit time scheduling problem $[(A,0), t]$ with $A = \{1,...,21\}$, 0 as given in Figure 2 and $t = 7$. Note that jobs 1,2,...,7 constitute a chain of length 7 in (A, 0).

(1) We run the modified choice verification algorithm for this problem as described in the proof of Lemma 3 and demonstrate that there is **no feasible schedule** for this problem.

1. The a priori bounds are $b(i) = i$ for jobs $i = 1,...,7$.
2. For the maximal elements 7, 15 and 21, the correct bound is 7.
3. For the jobs of dual height 2, i.e. for jobs 6, 14 and 20, the bound 6 is obtained w.r.t. the inner time bound 7 and all elements $i = 1,...,6$. Therefore 6 is the correct bound for these elements.
4. For the jobs of dual height 3, i.e. for jobs 5, 8, 9, 10, 11, 12, 13 and 19, the bound 5 is obtained w.r.t. the inner time bounds 6 and 7 and all elements $i = 1,...,5$. Therefore, 5 is the correct bound for these elements.
5. For the jobs of dual height 4, i.e. for jobs 4, 16, 17 and 18, the following happens. For job 4, the correct bound 4 is obtained. For jobs 16, 17 and 18, the consideration of the sets {2, 16}, {2,17} and {2,18} yields, for any of the involved inner time bounds 5, 6 and 7, the value 1. I.e. for jobs 16, 17 and 18, the associated bound will be smaller or equal to 1. As $b(1) \leq 1$ is true due to the established a priori bound, the bound contradiction $\{\alpha \in A \mid b(\alpha) \leq 1\} > 3$ is obtained. This means that there is **no feasible schedule** for this problem.

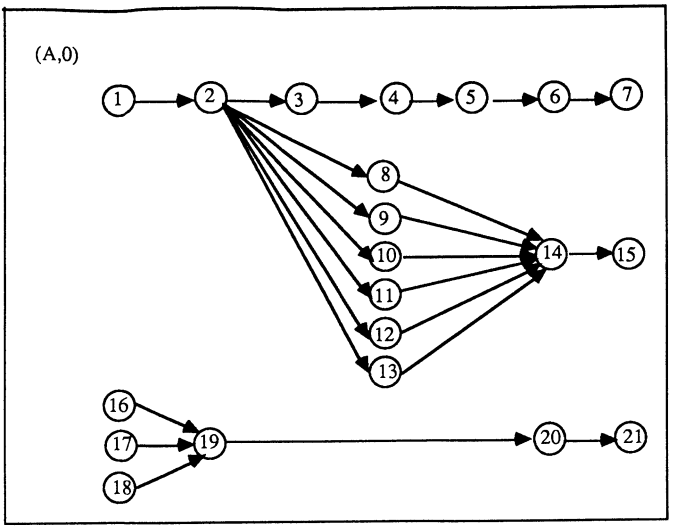


Fig. 2: A poset (A,0) with |A| = 21, containing a chain of length 7

(2) We **relax** (A,0) in Figure 2 by eleminating the relation (18, 19), i.e. job 18 will become an immediate predecessor of job 20. In this case, 18 is an element of dual height 3 and the corresponding bound will be 5. The correct bounds for jobs 16 and 17 would finally be 1 and the bound b(i) = i for i = 1,...,3 would also be confirmed. So, we would end up with the following situation:

b(1) = b(16) = b(17) = 1, b(2) = 2, b(3) = 3, b(4) = 4, b(5) = b(8) = b(9) = b(10) = b(11) = b(12) = b(13) = b(18) = b(19) = 5, b(6) = b(14) = b(20) = 6 and b(7) = b(15) = b(21) = 7. Obviously, there is no bound contradiction, i.e. there is a solution to this scheduling problem. A valid schedule corresponding to the given bounds is given by its Gantt-chart in Figure 3. Note that the schedule is **uniquely** determined up to permutations among jobs 8,...,13.

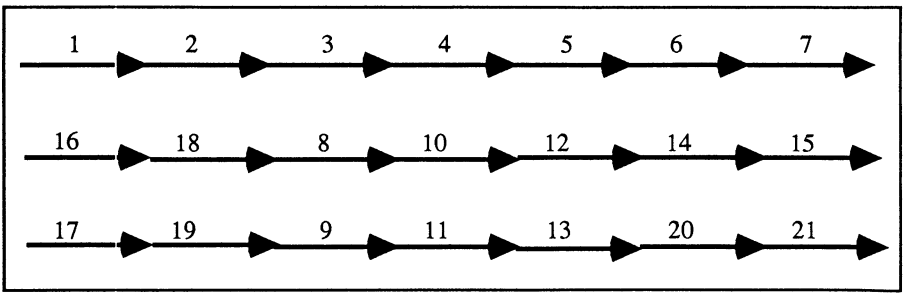


Fig. 3: A solution to a slight modification of the problem discussed in (1), represented by its Gantt-chart

We just mention that Lemma 4 (cf. [9]) yields the polynomial time solvability for 3 machine unit time scheduling problems [(A,0), t] for which (A,0) contains a chain of length greater or equal to t. This can be slightly extended to the existence of a chain of length t - r, r fixed, in (A,0). For an $O(n^{3+2r})$ algorithm to solve this type of problems cf. [1], [23].

Lemma 4 also shows a way to extend a solvable special case to the additional consideration of release and due dates. The extension for an m-machine problem [(A,0), t] would be based on the generalized normalization procedure, i.e. means the transition to an (m + 1) machine problem [(A´,0´), t] for which (A´,0´) contains a newly introduced chain $\beta_1,...,\beta_t$ that also serves as a means for the incorporation of release and due dates via order relations. The basic idea would again be to restrict M to those sets B with $1 \leq |B| < m + 1$ that contain an element β_j from the chain and that induce a bound smaller or equal to j - 1. In this situation, the ability to identify a test choice among the (at most m - 1 many) jobs different from β_j in B would be sufficient to yield a solution to the more general case involving release and due dates. I.e. a **general** polynomial solution for finding test choices for sets of at most m - 1 jobs (m machine case) leads to a solution for the m-machine case with release and due dates along the lines described in connection with Lemma 3. To put this more formally, we have the following:

LEMMA 6:

Any general $O(m^r)$ test choice construction procedure for sets of jobs of cardinality at most m - 1 leads to an $O(n^k)$, with $k = \max(m,r)$, solution algorithm for the m machine unit time scheduling problem (cf. the choice verification algorithm) and to an $O(n^{k´})$ with $k´= \max(m + 1,r)$ solution algorithm for the m machine unit time scheduling problem with **release** and **due** dates. This also holds for **m-profiles** instead of m machines!

Lemma 6 follows immediately from what was said above and has the nice feature that the consideration of release and due dates increases the exponent of the time bound by at most 1. If the time for the identification of test choices exceeds m, the additional consideration of release and due dates (and m-profiles) has no effect at all on the time bound. Lemma 6 might, for instance, be of great use in connection with special problem classes such as **interval orders** or **forests**, which are discussed next. For w.r.t. these two special classes, a polynomial time solution for the additional consideration of release and due dates is not yet known. However, the application hinges on the identification of an appropriate test choice finding mechanism, as the special mechanisms identified below for interval orders and forests without release and due dates do **not** apply to the extension in Lemma 6. So the requirement of a **general** test choice finding mechanism in Lemma 6 is essential in order to obtain the result.

We next consider the special class of so-called **interval orders**, i.e. posets (A,0) that have an interval order representation in the following sense: To each $\alpha \in A$ an interval $[u_\alpha, o_\alpha] \subseteq \mathbb{R}_+^1$ can be assigned, such that for $\alpha \neq \beta$ we have $(\alpha, \beta) \in 0 \Leftrightarrow o_\alpha \leq u_\beta$. Interval orders are of great importance in scheduling (cf. [19], [24], [25]), as any schedule naturally induces an associated interval order. There are a number of equivalent characterizations of interval orders [5], [11], [12], [25]. We mention here the characterization via the **forbidden** 4-element subposet given in Figure 4 and the following criterion:
(A,0) with $|A| = n$ is an interval order iff there is a ordering $\alpha_1, \alpha_2, \ldots, \alpha_n$ among the jobs such that $N(\alpha_j) \subseteq N(\alpha_j)$ holds whenever $i \leq j$. I.e. (A,0) is an interval order iff the set of successor-sets of jobs is totally ordered w.r.t. set inclusion. (This holds analogously for the sets of predecessors of jobs). Whether (A,0) is an interval order can be tested in $O(n^2)$, and if it is, an associated ordering of the jobs can also be given in that time. In the following, we assume jobs to be already numbered accordingly.

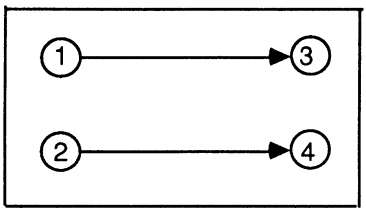


Fig. 4: Diagram of the forbidden subposet whose absence characterizes interval orders

LEMMA 7: (cf. also [22])

m-machine unit time scheduling of interval orders is possible in $O(n^2)$ time for any m.

Proof:

Note first, that the normalization to $|A| = m \cdot t$ by adding independent elements remains in the class of interval orders. As mentioned above, we can then assume within the time bound $O(n^2)$ that jobs are ordered w.r.t. decreasing sets of successors. Now given any considered set B with $1 \leq |B| < m$, we pick as a choice a job $\alpha \in A$ for which $N(\alpha)$ is maximal among all jobs in B. Obviously, this yields a **test choice** (which is obviously **non-blocking**). Even more α being a choice for some set B with $|B| \geq 2$ does not have any influence on the bound for α beyond the influence resulting from $B = \{\alpha\}$ anyway, as $N(\alpha) = \bigcup_{\gamma \in B} N(\gamma)$ will be true. Therefore, the computation can be restricted to sets B with $|B| = 1$, i.e. steps 2 to 4 in the choice verification algorithm for sets B with

$|B| = 1$ are sufficient to solve the problem. I.e. in $O(n^2)$ time, a decision is obtained, whether there exists a valid schedule and if it does, a schedule is determined in that time. Of course it is also immediately obtained, that **priority scheduling** w.r.t. the mentioned ordering of the jobs yields exactly the same solution as the one arising with the choice verification algorithm, as described above. In fact, that is the solution proposed in [22], which has the same time complexity. ■

The last special class of posets treated in this paper are **forests**. Forests are a natural generalisation of trees, which is preferred here for the following reason: while an application of the normalization procedure (adding independent jobs to a poset) leaves the class of trees, it stays within the class of forests. On the other hand, the known algorithm for trees [13], [14], [27] applies to forests as well. Concerning the treatment of forests, one distinguishes **inforests** and **outforests**. Inforests and outforests can be transformed into each other by a simple **dualization**. This holds similar for associated schedules. This allows to restrict the algorithmic treatment essentially to one of the two cases.

LEMMA 8: (cf. also [13], [14], [27])

m-machine scheduling of forests is possible in $O(n^2)$ time for any m.

Proof:

As mentioned above, one can w.l.o.g. restrict attention to the case of **outforests**. The classical approach in [14] then consists in determining in $O(n^2)$ time complexity the dual height of all jobs and then doing **priority scheduling** w.r.t. the priority ordering among the jobs, induced by increasing dual height of jobs; i.e. jobs of greater dual height are preferred. The correctness of this approach follows from a quite straightforward inductive argument. In our context, the argument is (in an intuitive description) as follows. W.l.o.g., one can start with m jobs. Let S denote a set of m minimal elements with lexicographically greatest dual height. Assume that starting with S does not lead to a valid schedule, while starting a set S' of minimal elements which are not lexicographically minimal does. In this case, we apply the inductive hypothesis to the subproblem arising after starting S'. This leads to a schedule for the subproblem that arises from priority scheduling w.r.t. the dual height. It is not difficult to see that within this schedule, successors of elements of "low" dual height in S' will w.l.o.g. not appear until all elements from S are scheduled. Here the main observation is that for **outtrees**, after finishing a job from S, a successor of this job becomes

available, whose dual height is still greater on equal to the dual height of those jobs with "low" dual height in S'. Seeing this, we can exchange each job from S, scheduled later, with a job of "low" dual height, scheduled in the first time slot. This yields a valid schedule, starting with S, a contradiction to our assumption.

Of course this result solves the problem in $O(n^2)$ time via **priority scheduling** w.r.t. the dual height. It is illustrative to observe, however, that this result also yields a (**non-blocking**) test choice by chosing within a set B of cardinaliy $1 \leq |B| < m$ a job with greatest dual height. The proof of this statement is as follows: If the problem has no solution, every choice will lead to a bound contradiction for the associated bounds, due to the Representation Theorem. If there is a solution, priority scheduling w.r.t. the dual height will yield a valid schedule, due to the argument given above. W.l.o.g. we can assume that w.r.t. this schedule, the chosen element for a set B with $1 \leq |B| < m$ (which has maximal dual height in B) will be scheduled earliest. As argued in connection with the Representation Theorem this is enough to prove that the schedule is compatible with the choice-induced bounds. This in turn guarantees that no bound contradiction will occur. Thus the test-choice property is established.

It is interesting to note that the determination of the test-choice depended bound may, in case there is a solution, lead to a different kind of priority schedule then priority scheduling w.r.t. the dual height, cf. Example 9 for this phenomenon. I.e. in the sense of the Representation Theorem, priority scheduling according to the dual height may not necessarily lead to the **lexicographically smallest** start set w.r.t. the test-choice induced bound. However, both ways of scheduling lead either to valid schedules or establish the non existence of a solution, a potential coincidence that is included in the formulation of the Representation Theorem. Needless to say that, from a complexity point of view, $O(n^2)$ priority scheduling w.r.t. the dual height is much superior then the $O(n^m)$ priority scheduling w.r.t. the choice-induced bounds. ■

EXAMPLE 9:

Consider the 3-machine unit time scheduling problem $[(A,0), t]$ with $A = \{1,...,18\}$, 0 an out-forest as given in Figure 5 by a diagram and $t = 6$.

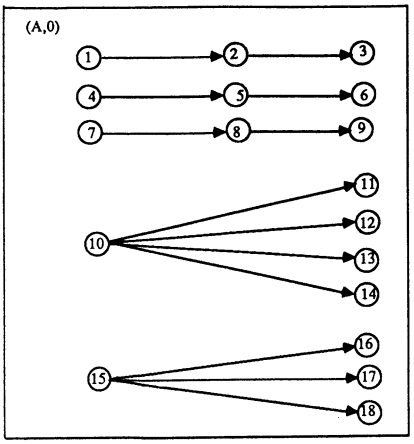


Fig. 5: An outforest on a base set of 18 elements

A priority list on the non-maximal elements of (A,0) compatible with the dual height of jobs, is $1 < 4 < 7 < 10 < 15 < 2 < 5 < 8$. A corresponding feasible schedule for the problem is given by a Gantt- chart in Figure 6a.

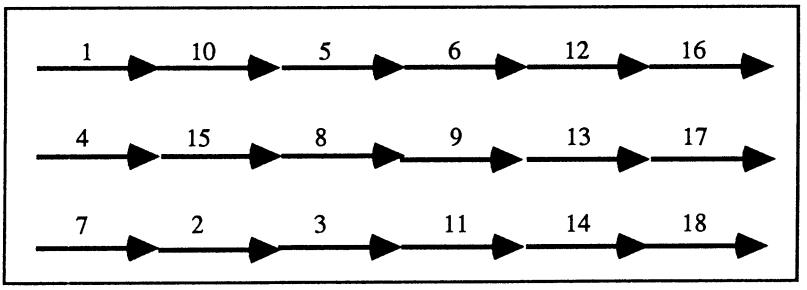


Fig. 6a: A Gantt chart for a dual height-induced schedule

Next we consider the (non-blocking) choice on sets of two elements, induced by the dual height of jobs in the way stated above. As an application of the choice verification algorithm, we obtain the corresponding bounds for all non-maximal elements as $b(15) = 4$, $b(1) = b(4) = b(7) = 4$ and, finally, $b(10) = 3$. A corresponding feasible schedule for the problem is given by a Gantt-chart in Figure 6b. Note that this schedule is **not** compatible with priority scheduling w.r.t. the dual height of jobs, which requires the set $\{1, 4, 7\}$ of jobs as start set.

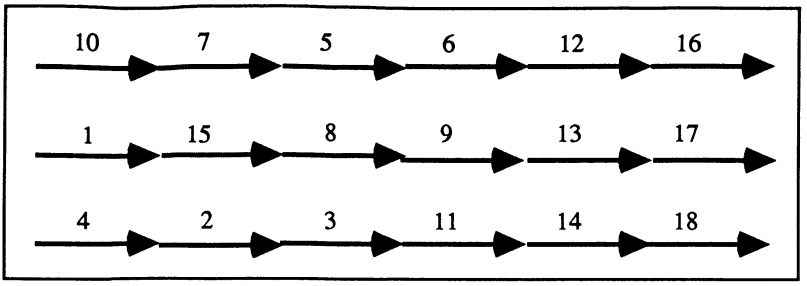


Fig. 6b: A Gantt chart for a bound-induced schedule

Notice, finally, that the schedule given in Figure 6a, also fulfills the statement of the Representation Theorem in the sense that after removing jobs {1,4,7}, there is no bound contradiction on the remaining jobs w.r.t. the bounds t - 1, and so forth.

To close this Section, we mention that there are some other nice special cases solved like **opposing forests** for m arbitrary, cf. [4], **posets of bounded height** r for arbitrary fixed m with $O(n^{r(m-1)+1})$ time complexity, cf. [3], and, finally, **posets of bounded width** r for m arbitrary with $O(n^{2r})$ time complexity, cf. [1], [23]. This last mentioned algorithm consists in dynamic programming w.r.t. the **so-called fronts** (i.e. minimal sets of not yet scheduled jobs) that may be encountered during on-line scheduling. The number of such fronts is bounded by n^r for posets of bounded width r. Note that membership in all three special classes can be verified efficiently. (For another reference to posets of bounded width see Theorem 10 (3) below).

We mention finally, that up to now no definite insight (as discussed in this Section for some other special cases) is available for possible connections between these special cases and the corresponding choice-test problem in the sense of the Representation Theorem. The same is partly true w.r.t. the possible incorporation of release and due dates and, furthermore, m-profiles. However, there is presently work going on concerning these questions at the University of Passau.

4. TOWARDS A POLYNOMIAL TIME ALGORITHM FOR GENERAL M-MACHINE UNIT TIME SCHEDULING PROBLEMS

A major insight from the Representation Theorem and the choice verification algorithm is, that somehow a solution to the choice problem for sets $1 < |B| < m$ is the main conceptual - and probably also algorithmic-step to make in order to obtain a general algorithm. However, while we saw in Section 3 that this may be done quite nicely in special cases, there is, unfortunately, not much insight available up to now for this problem in the general case.

Therefore, we will take a slightly different route in this Section (where we will omit the normalization condition $|A| = m \cdot t$). We will try to accumulate (in polynomial time) information concerning the placing of all jobs. The available information will be collected in the form of a **bounding system**, that gives **lower** and **upper** bounds $L(\alpha)$ and $U(\alpha)$ for all elements $\alpha \in A$. In this bounding system, we will particularly incorporate bounds for sets B with $1 < |B| < m$ in those cases, where **at most one** element from B can accept the corresponding lower or upper bound $L(B)$ or $U(B)$ for B without causing a bound contradiction (so-called **forced choices**, cf. the **validation step** described below). In order to get an intuitive idea of this last point, compare Example 1 (2). Here, at a certain point, we found - using the bounds for single elements - the set bound $b^*(\{1,8\}) = 1$. As job 8 has predecessors and therefore cannot be finished by time 1, we could deduce the bound 1 for job 1. I.e. here, the **choice** on the set $\{1,8\}$ is **forced**. Actually, later on a contradiction was obtained in this example, using this observation, i.e. there is no solution at all in the example.

When talking about a bounding system, we will require it to be **valid** and **monotone** in the following sense: If there are feasible schedules T for the problem, then for all $\alpha \in A$ and all such T, $L(\alpha) \leq T(\alpha) < T(\alpha) + 1 \leq U(\alpha)$ must be true. The same holds analogously for bounds of sets of higher cardinality. Call a bounding system **monotone**, if adding new relations or further elements or decreasing t will not decrease lower and not increase upper bounds; again also w.r.t. sets instead of elements. Call a bounding system **not contradictory** if $0 \leq L(\alpha) < U(\alpha) \leq t$ holds for all $\alpha \in A$ and if for all $0 \leq i < j \leq t$, no bound contradiction of the form $|\{\alpha \in A \mid i \leq L(\alpha) < U(\alpha) \leq j\}| > m(j - i)$ occurs. Obviously, given a valid bounding system, this has to be not contradictory in order to allow the existence of a valid schedule for the problem. Our hope is that the contrary is also true, i.e. as a kind of **WORKING-CONJECTURE**, we hope for the existence of a bounding system, deducable in polynomial time, such that the bounding system not being contradictory guarantees the existence of a feasible schedule for a given problem.

THEOREM 10:

If the working-conjecture is correct for a bounding system (for the m-machine case), and if the bounding system is computable in $O(n^k)$ time ($k \geq 2$), then the general m-machine unit time scheduling problem can also be solved in polynomial time. In particular, the following time bounds can be established:

(1) The existence of a solution for the problem can be decided in $O(n^k)$ time:

(2) If a solution exists, a feasible schedule can be obtained in $O(n^{k+2})$ time.

(3) For posets of bounded width, the feasible schedule in (2) can even be constructed in $O(n^{k+1})$ time.

Proof:

(1) The algorithm consists in the repetition of a **basic step**, which essentially consists in the computation of the bounding system for individual jobs for [(A,0),t] in $O(n^k)$ time. In $O(n^2)$ time it is then checked, whether this system is contradictory. If it is, the problem has no solution, otherwise it has, due to the Working Conjecture. This solves (1).

(2) Assume that there is a solution for [(A,0),t]. As we will demonstrate below in Examples 12 to 16, the bounds for jobs obtained from the computation of the bounding system in (1) may be informative w.r.t. finding a valid schedule. However, we will also demonstrate in Example 14 that, in general, additional computational effort is needed. A straightforward way to proceed is to add a further relation (α, β) with $\alpha \neq \beta$ and α θ-independent from β to (A, 0) and check with the basic algorithm in (1), whether the resulting problem [(A, 0´), t] is solvable. If it is not, we substitue (α, β) by a different relation. Otherwise, we proceed from [(A, 0´), t] as described for (A, 0). After at most $O(n^2)$ steps, we arrive at a problem [(A, 0*), t], which has a solution, but for which any proper order extension of 0* does no longer allow a solution. It is easily checked that each such poset 0* consists of a linear sequence of at most t blocks of at most m independent elements each. This sequence corresponds bijectively to a feasible schedule, that is thus obtained in $O(n^{k+2})$ time.

(3) Another way to produce a valid schedule in (2) is due to the following consideration. As the problem has a solution, certain minimal elements α of (A, 0) can start at time zero. In the non-trivial case, there are more minimal elements available as machines. So, the question arises, which jobs to start. It is not difficult to prove, that the possibility to start a job α is equivalent to the statement, that the partial order (A, 0´) with $0' = \{0 \cup \{\alpha, \beta\} \mid \beta \text{ not minimal in } (A, 0)\}$ has a solution. So we could consider some minimal element α and check the existence of a solution for the associated poset (A, 0´) in $O(n^k)$ time. If the resulting problem has no solution, we try the next minimal element in (A, 0). Otherwise, we proceed from (A, 0´) with the next minimal element in (A, 0´) (which are the same minimal elements as those in (A, 0)). By proceeding through the minimal elements, we will at some point have identified m minimal elements which iteratively lead to solvable problems. It is immediate that these m elements can be scheduled at time zero and allow a continuation to a feasible schedule by applying the same procedure to the resulting subproblems. In proceeding this way, a $O(n^{k+1})$ procedure has to be applied at most $n \cdot t / m$ times. This yields again the $O(n^{k+2})$ time bound, also obtained in (2). However, for posets of **bounded width** $r \in \mathbb{N}$, the number of minimal jobs to be considered in each time slot is bounded by the constant r.

(Note, that the basic poset extension operation, discussed in (2), does not increase the width!) Therefore, the time complexity for finding a valid schedule can be reduced to $O(n^{k+1})$, as stated in the theorem. ■

In the following, we introduce a **bounding system** that can be determined in $O(n^5)$ time and that allows a number of insights in a variety of illustrative examples, some of which are treated below and in Section 5 of this paper. The given boundary system seems to be the natural core to start with for m-machine scheduling. It is certainly not sufficient for general $m \in \mathbb{N}$, cf. Example 16 below. All studied examples so far indicate however, that the Working-Conjecture might apply to this bounding system for the 3-machine case. A report on some progress concerning a correctness proof for this statement is given in Section 5. On the other hand, we give in Example 14 an indication on to how the bounding system might be tightened (still in polynomial time), if the given bounding system should turn out to be insufficient even in the 3-machine case.

BOUNDING SYSTEM

The system gives valid and monotone **lower and upper bounds** $L(\alpha)$ and $U(\alpha)$ on the starting and completion times of jobs (and sets of jobs) in $O(n^5)$ time. For a set B of jobs, a lower (upper) bound L(B) (U(B)) means that at least one job in B has to start not earlier than L(B) (has to be completed by U(B)). In addition, bounds are checked for **consistency**.

(1) The basic step and the associated closure operator

Let an m-machine unit time scheduling problem [(A, 0), t] be given, together with present lower bounds $0 \le L(\alpha) < U(\alpha) \le t$ for all $\alpha \in A$, starting originally form $L(\alpha) = 0$ and $U(\alpha) = t$ for all $\alpha \in A$. We assume that the present bounds are consistent in the sense that $|\{\alpha \in A \mid i \le L(\alpha)$ and $U(\alpha) \le j\}| \le m \cdot (j - i)$ for all pairs i, j with $0 \le i < j \le t$. Then we update $U(\alpha)$ by exactly the basic formula used in Sections 2 and 3 w.r.t. the bounds $b(\alpha)$. (This holds similarly again for sets B with more than one element, where we now however consider only one such set at a time w.r.t. the bounds $b(\alpha)$ obtained, as described, for single elements α). The bounds U(B) and L(B) for $|B| > 1$ are defined analogously, i.e. via **dualization**. We first determine $U(\alpha)$ by backward recursion and afterwards $L(\alpha)$ by forward recursion for all $\alpha \in A$. Both definitions do not infere. In $O(n^2)$ time, all these values are obtained. In the same time complexity, it can be checked whether $U(\alpha) \le L(\beta)$ holds for pairs α, β of θ-independent elements. If this is the case, we add all the correspondent relations (α, β) to (A, 0) to obtain an extension (A, 0'). Obviously, any solution for

[(A, 0), t] must be a solution for [(A, 0´), t], too. So as part of a **closure operation**, we can proceed with (A, 0´) and determine the corresponded bounds as part of the described basic step. Note that a basic step needs $O(n^2)$ time and has to be repeated at most n^2 times.

(2) The validation step and the associated closure operator

If we stop with step (1), the question arises whether a job α with bounds $L(\alpha)$ and $U(\alpha)$ may really start at time $L(\alpha)$ or complete at time $U(\alpha)$ without causing a contradiction w.r.t. the bounds imposed on the other jobs. We can check this **local validity** of the obtained bounds by first fixing the placing of job α into the time slot $[L(\alpha), L(\alpha) + 1)$ by putting (tentatively) $U'(\alpha) := L(\alpha) + 1$ and afterwards by fixing the placing of job α into the time slot $[U(\alpha) - 1, U(\alpha))$ in an analogous way. As we will demonstrate below, this kind of fixing may lead to a bound inconsistency, because (e.g. w.r.t. the upper bound $U(\alpha)$) for certain time windows [i, j] with $L(\alpha) \leq i < U(\alpha) < j$, a set $A_{i,j}$ of jobs that are not successors of α, must be placed within this time window, for which the number $|A_{ij}|$ exceeds $m(j - i) - |\{\gamma \in N(\alpha) \mid b(\gamma) \leq j\}| - 1$; in this context, on can w.l.o.g. restrict attention to the case where A_{ij} consists only of such jobs, that are independent of α. In this described case, $U(\alpha)$ must be reduced at least to i, which in particular means that the order relations (α, β) with $\beta \in A_{ij}$ arbitrary, might be added to the present partial order to obtain an extension similar to (1). In this case, one can as part of a **closure operation** again proceed with (1), afterwards again with (2) and so forth. Note that a validation step needs $O(n^3)$ time, as it actually means the performance of up to 2n basic bound computations and consistency checks. The **complete closure operator** resulting from (1) and (2) consists thus in at most n^2 applications of either an $O(n^2)$ or $O(n^3)$ procedure, i.e. can be done in $O(n^5)$ time complexity. I finally stops in a "**closed situation**" in the sense that the bounds obtained so far do not lead to a further tightening of the bounds.

A major aspect of the validation step is the incorporation of **forced choices** for sets B with $1 < |B| < m$ in $O(n^5)$ time, i.e. without scanning all sets B of cardinality at most m - 1. This independence of m, which is demonstrated in the next lemma, is nice from a computational point of view. However, given the NP-completeness of m-machine unit time scheduling for m arbitrary [18], [28], is means of course the insufficiency of the given bounding system in connection with the Working-Conjecture w.r.t. general $m \in \mathbb{N}$. The case of m = 3 might, of course, be different. More comments on that follow below.

LEMMA 11:

Consider an m-machine unit time scheduling problem [(A, 0), t] for which the associated bounding system is not contradictory. Then for every set B with $1 < |B| < m$, there are at least two elements $\alpha_1, \alpha_2 \in B$ and at least two elements $\beta_1, \beta_2 \in B$ fulfilling $L(\alpha_i) \leq U(B) - 1$ and $U(\beta_i) \geq L(B) + 1$ for i = 1,2. I.e. all forced or impossible choices for considered sets B (i.e. situations, where at most one element α, β from B fulfills $L(\alpha) \leq U(B) - 1$ or $U(\beta) \geq L(B) + 1$) are already incorporated (via $U(\alpha) \leq U(B)$ or $L(\beta) \geq L(B)$) into the bounding system, due to the **validation step**.

Proof:

Assume that at some point in the computation of the bounding system, only one job α or β from a (currently) independent set B with $1 < |B| < m$ fulfills $L(\alpha) \leq U(B) - 1$ or $U(\beta) \geq L(B) + 1$. We concentrate w.l.o.g. on the first case. Put $i := \min \{L(\gamma) \mid \gamma \in B \setminus \{\alpha\}\} \geq U(B)$, chose j as a defining time point for U(B), i.e. such that $U(B) = j - \lceil |\{\delta \in \bigcup_{\gamma \in B} N(\gamma) \mid b(\alpha) \leq j\}| / m \rceil$ with $j > i$ holds.

Now consider the **validation step** for $U(\alpha)$ and the time window [i, j], where, because of the basic algorithmic step and the given independence of B, $U(\alpha) \geq i + 1 \geq U(B) + 1$ holds. Here, $j > U(\alpha)$ can also w.l.o.g. be assumed, as otherwise a bound contradiction w.r.t. the set $B \setminus \{\alpha\}$ would be obtained. The validation step then yields $U(\alpha) \leq i$ and leads to the introduction of all relations (α, γ) with $\gamma \in B \setminus \{\alpha\}$ to the poset, followed by the establishing of $U(\alpha) \leq U(B)$, i.e. to the forced choice, as stated. This solves forced choice situations. Obviously, a modification of this argument works for impossible choices, i.e. yields a bound contradiction, should **no** job from a set B be able to accept the corresponding bounds L(B) and U(B). ■

In the following, we apply the bounding system approach to a number of interesting 3-machine unit time scheduling problems. Note that in connection with Theorem 10, which uses an appropriate introduction of new relations, we are using a **combined $O(n^7)$ algorithm**, for which there is some evidence that it might actually solve the 3-machine case completely.

EXAMPLE 12:

Consider the following 3-machine unit time scheduling problem [(A, 0), 5], where (A, 0) is given by a diagram in Figure 7.

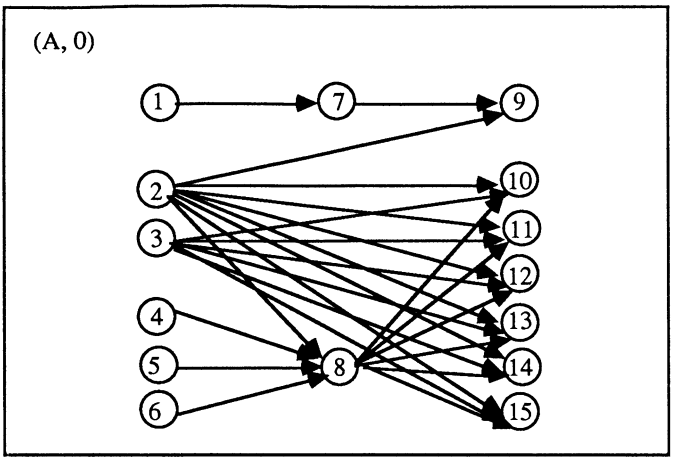


Fig. 7: A poset with unique starting set in an optimal 3-machine unit time schedule

The problem was chosen, because it has a solution in which, furthermore, the associated schedule is essentially unique. As the problem has a solution, no bound contradiction can occur. However we will see, that the bounds obtained directly lead to the solution.

The basic step in the computation of the considered bounding system yields U(2) = U(3) = U(4) = U(5) = U(6) = 2, U(8) = 3 and L(9) = 2. As a consequence, we can add the additional relations (4, 9), (5, 9) and (6, 9) to the poset. These leads to the modification L(9) = 3, yielding the introduction of the additional relation (8, 9). As a consequence, we obtain U(8) = 2 and thus U(4) = U(5) = U(6) = 1. The first two time slots have therefore to be filled by jobs {4,5,6} and {2,3,4}, respectively. As part of the validation step, L(1) = 2, U(1) = 3, L(7) = 3, U(7) = 4, L(9) = 4, U(9) = 5 are obtained, together with L(i) = 2 and U(i) = 5 for all jobs i = 10,...,15. This determines the solution to the problem, which is **unique** up to permutations among the jobs 10,...,15. Note that the bounds obtained in this example exactly give the range, in which the respective jobs might be placed in valid schedules for the problem.

EXAMPLE 13:

Consider the following (3-critical) unit time scheduling problem [(A, 0), 4] with |A| = 12 and (A,0) as given in Figure 8 by a diagram. This problem has a solution in 5 time units, however not in 4 time units.

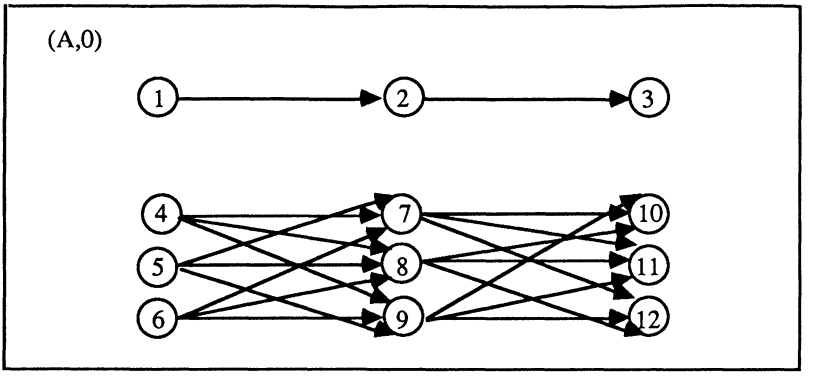


Fig. 8: A 3-critical poset with an optimal schedule length of 5 time units

The basic step in the computation of the considered bounding system for $t = 5$ yields $U(1) = U(4) = U(5) = U(6) = 2$ and $L(3) = L(10) = L(11) = L(12) = 2$. As a consequence, we can introduce the additional relations (1,10), (1,11), (1,12) and (4,3), (5,3) and (6,3). Annother application of the basic step then yields $U(4) = U(5) = U(6) = 1$ as well as $L(10) = L(11) = L(12) = 3$. As part of a validation step, $L(1) = 1$ and consequently $L(3) = 3$ will be obtained. This yields the bound contradiction $|\{\alpha \in A \mid L(\alpha) \geq 3\}| > 3$, i.e. there is no solution in 4 time units.

EXAMPLE 14:

Consider the following 3 machine unit time scheduling problem [(A, 0), t], where (A, 0) is given by a diagram in Figure 9.

(1) The basic step in the computation of the considered bounding system yields $U(10) = U(11) = 5$, $U(8) = U(9) = 4$ and $U(i) = 3$ for $i = 1,...,6$. It is easily seen that all these bounds are really obtained in valid schedules, so they cannot be improved. In particular, the validation step and the consideration of jobs B with $|B| = 2$ as part of the computation of the bounding system will not further decrease these bounds.

(2) It is interesting to note, however, that because of $L(10) = L(11) = 2$, the consideration of the set $B = \{10,11\}$ with associated bound $U(B) = 3$ (which can be fulfilled by both jobs and thus does not lead to the introduction of new relations) implies that either jobs $\{1,2,3\}$ or jobs $\{4,5,6\}$ have to be chosen as starting set. This cannot immediately be deduced from the considered bounding system, i.e. additional steps as described in Steps 2 and 3 in Theorem 10 are needed here, cf. also (4) below. Note, however, that it could be deduced here that necessarily $L(7) \geq 1$ holds. This sharper bound could have even been obtained within the same time complexity. I.e. we see, that the bounds resulting from the considered bounding system are usually **not sharp**. Also we see, how they might be **tightened** if the Working-Conjecture does not hold w.r.t. the considered bounding system even in the case of $m = 3$.

(3) Along the lines followed in (2), it can be shown that jobs 8 and 9 will either be scheduled at time 2 or at time 4 or 5 in a feasible schedule for the problem. I.e. the possible time points, where jobs may be placed in feasible schedules, are, in general, **no intervals**.

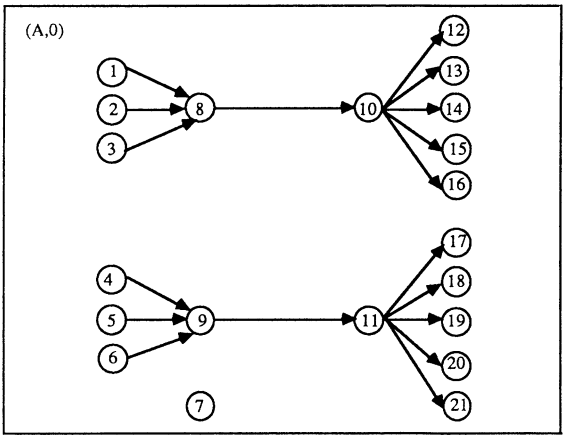


Fig. 9: A poset with an optimal 3-machine unit time schedule length of 7 time units

(4) If we apply the procedure described in connection with Theorem 8 in order to find a feasible schedule for the problem, we might first try to start job 7 at time 0 (which is impossible) by adding the additional order relations (7, i) with $i = 8,...,21$. A bound contradiction would then, however, be obtained as follows. Because of the additional relations, $L(8) = L(9) = 2$ and thus $L(10) = L(11) = 3$ holds. Obviously, this contradicts $U(\{10,11\}) = 3$, which would lead to a bound contradiction via the validation step.

(5) If, instead, we try in (4) to start job 1 at time 0 (which is possible) by adding the additional relations (1, i) with $i = 8,...,21$ and would afterwards continue with the idea to start job 4 at time

0, too, (which is impossible then), we would obtain (after adding the additional order relations (4, i), i = 8,...,21) exactly the same bound contradiction as given in (4).

EXAMPLE 15:

Consider the following 3 machine unit time scheduling problem [(A, 0), 26], where (A, 0) is given by a diagram in Figure 10a. Note that the problem, that consists in the treatment of two parallel chains of independent elements is quite involved; cf. also Section 5 for this kind of structure. As the problem has a solution, no bounding contradictions will occur. The example was obtained as a counterexample for inductive proofs for certain (false) conjectures in the field. The main point is that the 12 independent minimal jobs 43,...,54 have to be scheduled within the first 4 time slots. This can be obtained from the considered boundary system as follows:

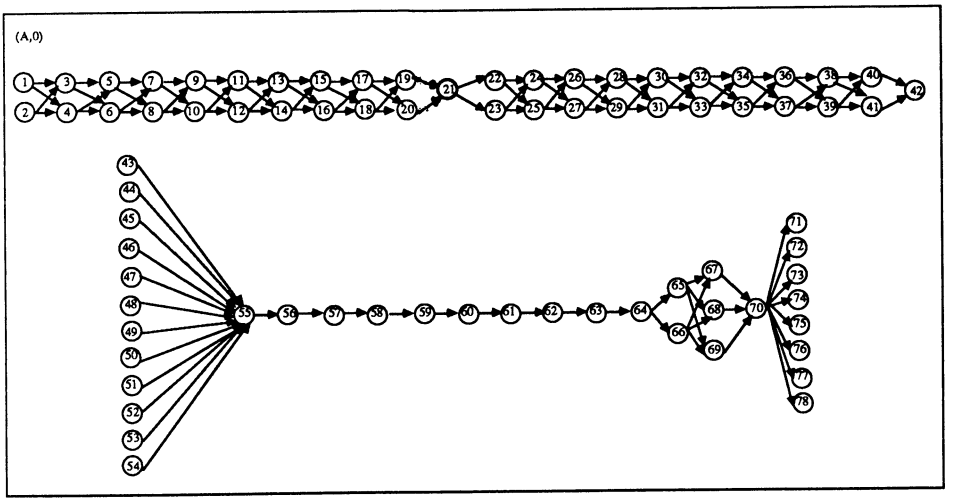


Fig. 10a: A poset (A, 0) with |A| = 78, that allows unit time scheduling on 3 machines in 26 time units. In any optimal schedule, jobs 43,...,54 have to be scheduled within the first 4 time slots.

(1) We first obtain L(21) = 10 and U(43) = ... = U(54) = 10. This will lead to the introduction of the new ordering relations (i, 21) with i = 43,...,54 and the modified lower bound L(21) = 11.

(2) Given (1), i.e. $L(\alpha) \geq 12$ for all successors of job 21, a basic step w.r.t. the originally obtained bound U(56) = 12 leads to the modification U(56) = 11, implying U(55) = 10.

(3) Given (2), a validation step for the originally obtained bound L(17) = 8 yields, that because jobs 55 and 17 (or 18) together have 28 predecessors and job 56 has to be finished by time 10, the new bounds L(17) = L(18) = 10 are obtained. This immediately implies L(21) = 12.

(4) A validation step for U(59) = 15, which was originally obtained, leads because of L(21) = 12 and a total number of successors of jobs 59 and 21 of 40 to the updated upper bound U(59) = 12, implying U(55) = 8.

(5) A validation step for the original bound L(11) = L(12) = 5 now yields, given that U(55) = 8, the adapted values L(11) = L(12) = 8. This directly implies L(21) = 13.

(6) A validation step for the original bound U(62) = 18, given that L(21) = 13, yields U(62) = 13. This implies U(56) = 7.

(7) A validation step for the original bound L(7) = L(8) = 3, given that U(56) = 7, yields L(7) = L(8) = 7 and therefore leads to the introduction of the new relations (56, 7) and (56, 8). This implies U(56) = 6 and consequently U(i) = 4 for i = 43,...,54. This essentially solves the problem.

A complete determination of the associated bounding system gives sharp bounds for the placing of all jobs. A solution, which is essentially unique, is given subsequently by the sequence of sets of 3 jobs for all time slots 1,...,26:

43	46	49	52	1	3	5	7	9	11	13	15	17	19	21	22	24	26	28	30	32	34	36	38	40	42
44	47	50	53	2	4	6	8	10	12	14	16	18	20	65	23	25	27	29	31	33	35	37	39	41	77
45	48	51	54	55	56	57	58	59	60	61	62	63	64	66	67	68	69	70	71	72	73	74	75	76	78

Fig. 10b: A feasible schedule for Example 15.

Examples 12 to 15 have shown the potential of the used bounding system for a number of 3-machine unit time scheduling problems (for further examples cf. Section 5). As already mentioned, there is some hope that this procedure with $O(n^7)$ time complexity really solves the 3-machine case (or at least may be extended towards a sufficient bounding system along the lines described in connection with Example 14). Approaches towards a proof of this statement will follow in Section 5. Before, we will indicate, where problems with this bounding system may arise for $m = 3$ and have to arise for increasing values $m \in \mathbb{N}$. As we will demonstrate in the concluding example of this Section, the problems are of the following type: It may well be that, for all sets B of jobs with $1 < |B| < m$, at least two jobs from each set B may be able to accept the bounds L(B) and/or U(B). The question still is, whether all these requirements can be met **simultaneously**. I.e. the question is, whether a particular start set can satisfy all requirements of

typ U(B) = 1 for all sets B considered, and if it can, whether a set of jobs for the second time slot can be found, that together with the starting set fulfills all requirements of the form U(B) = 2 and so forth. Incorporating checks of the type described into the determination of a bounding system might be the right way to extend the approach followed in Section 4 to greater values of m. Here, the need to check sets of m minimal elements concerning the described behaviour would cause the increase in time complexity to an at least **linear** dependence on m, that has to be expected.

EXAMPLE 16:

The example is essentially obtained from the reduction of CLIQUE to the m-machine unit time scheduling problem with m arbitrary and t = 3 that is given in [18]. We start here from the graph G, given in Figure 11a and transform it into the unit time scheduling problem [(A, 0), 3], with m = 7 and (A, 0) as given by a diagram in Figure 11b. The graph has no clique of cardinality 3 which will mean, that the scheduling problem has no solution.

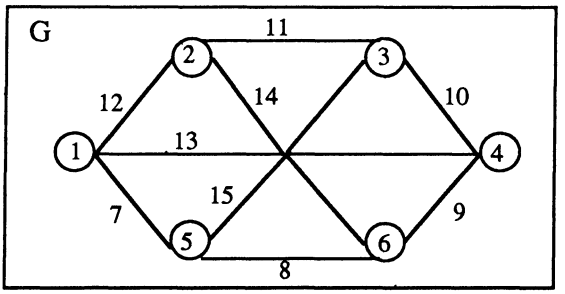


Fig. 11a: A graph G without cliques of 3 elements (so-called $K_{3,3}$)

Note that any job from {1,...,6} (which correspond to nodes of G) has exactly 3 successors, while any job from 7,...,15 (which correspond to vertices of G) has 2 predecessors, viz. the nodes that are incident with the associated vertex. For reasons of symmetry it is easy to see, that the associated bounding system will correctly place jobs 16,...,21 and will lead to the bounds $L(\alpha) = 0$ and $U(\alpha) = 2$ for $\alpha = 1,...,6$ and $L(\alpha) = 1$ and $U(\alpha) = 3$ for $\alpha = 7,...,15$. It is then easily obtained that particularly interesting θ-independent sets B in the sense of the Representation Theorem and Corollary 2 are as follows: they contain job 20, furthermore a set C of three jobs from {1,...,6} and finally, at most 2 further jobs from {7,...,15}. Here, it is easily obtained from the construction, that the latter jobs are determined by C in the sense that, given any set B as considered, there are at least 7 successors in {7,...,15}, due to the fact, that G contains no cliques,

i.e. 3 edges of G are incident with at most 2 vertices from G. Obviously, this implies the bound $U(B) = 1$ for all such sets B. Though in all cases, every element in B (with $|B| = 3$) can individually accept the upper time bound 1 because of $L(\alpha) = 0$ for all $\alpha \in \{1,...,6\}$, this does not mean, that all these requirements can be satisfied **simultaneously**. To the contrary, whatever 3-elementary set of jobs from $\{1,...,6\}$ is started at time 0, there remains a subset B with $|B| = 3$, from which no element was chosen. This shows the **non-existence** of a solution for the considered 7-machine unit time scheduling problem. [Note that similar examples have up to now not been found for the 3-machine case. Therefore, the $O(n^7)$ procedure resulting from the given bounding system and Theorem 10 might still be sufficient to solve this case].

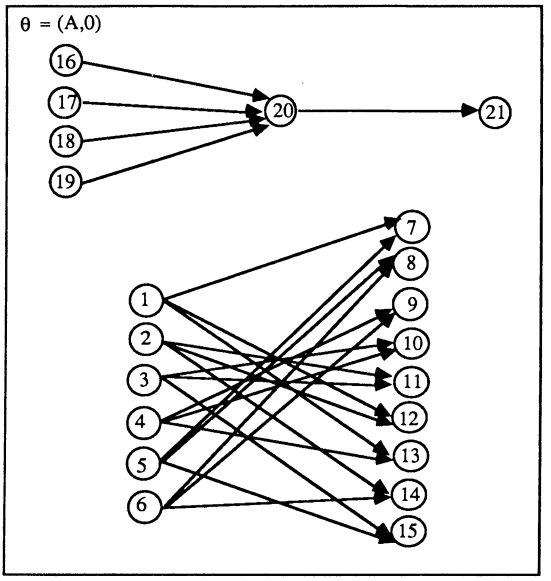


Fig. 11b: The poset (A, 0), essentially obtained from G via a standard reduction form CLIQUE.

5. M-CRITICAL POSETS

In Section 4, the question was raised of how to prove the intended result that a certain bounding system solves an m-machine unit time scheduling problem in the sense that if no bound contradiction is obtained, there is a valid schedule. Certainly, this question arises particularly in connection with the proposed bounding system for the 3-machine case. Such proof can be expected to be laborious, given the involved recursive nature (as closure operator) of the proposed bounding system already for the case $m = 3$.

In the following we try to follow an approach, that was already successful in other combinatorical optimization problems, viz. the concentration on special **critical** problem instances. The hope is that such critical instances may have a special structure and that this structure can be exploited.In addition, their efficient characterization would yield a proof of membership of the considered problem in CO-NP.

DEFINITION:

Call a poset $\theta = (A, 0)$ **m-critical**, if an optimal unit time schedule for this problem on m machines needs t time units, while after removing any element from 0 (or A), an optimal schedule needs only t - 1 time units.

Examples of 3-critical posets were already considered, cf. Examples 1 and 13. We show next, that for a proof concerning the correctness of a valid and monoton bounding system for m-machine unit time scheduling, only the instances $[(A, 0), t - 1]$, where (A, 0) is m-critical and t is the optimal schedule length for (A, 0), need to be considered.

THEOREM 17:

Assume for a valid and monotone bounding system for m machine unit time scheduling, that it yields a bound contradiction for all problems instances $[(A, 0), t - 1]$, where (A, 0) is m-critical with optimal schedule length t. Then the bounding system is correct, i.e. it yields a bound contradiction for some arbitrary problem instance iff there is no solution to the problem.

Proof:
As the bounding system is feasible, any bound contradiction implies that the respective problem instance has no solution.

Assume, for the other direction, that there is no bound contradiction for a problem instance $[(A,0), t]$. Then one has to show that there is a solution to the problem. We prove this by induction on 0, regarded as a set of pairs. The case of a trivial partial order (only independent elements) follows immediately. The induction step is then as follows. We assume that $[(A, 0), t]$ has no solution. W.o.l.g. we can assume that $[(A, 0, t + 1]$ has a solution, for otherwise, due to the **monotonicity** behaviour of the bounding system, there would be certainly no bounding contradiction w.r.t. $t + 1$ and t could thus be replaced by $t + 1$ in the argument. By the induction hypothesis, we know the correctness of the bounding system after eleminating any

relation or any element from (A, 0), leading to a poset (A´, 0´) with $A' \subseteq A$ and $0' \subset 0$. Again, relying on the monotonicity of the bounding system, there will certainly be no bound contradiction resulting from an application of the bounding system to [(A´, 0´), t]. Because of the induction hypothesis, this implies that (A´, 0´) can indeed be scheduled in t time units. Thus we have obtained a problem instance [(A, 0), t], for which there is a solution in t + 1 but not in t time units and for which any relaxation of (A, 0) allows a solution in t time units. I.e. we have obtained an **m-critical** poset (A, 0), for which the bounding system produces a bound contradiction by assumption. This is a contradiction to the behaviour required in the theorem, concluding the proof. ■

Because of Theorem 17, the consideration of m-critical posets is sufficient in proofs of the correctness of valid and monotone bounding systems as considered in Section 4. In a certain sense (cf. e.g. the proof of Theorem 17) they constitute the most difficult problem instances to deal with. Still, there is some hope that such "extreme" problem instances might have such a special structure, that this may really facilitated a proof. This idea was e.g. carried out in [26] and was, at least partially, successful. The major step in this direction was the subsequently formulated Theorem 19, that characterizes the **2-critical posets** exactly as the class of so-called **odd weak orders**, where $\theta = (A, 0)$ is called an odd semi-order, if θ consists of a chain of autonomous subposets, each of which consists of an uneven number of independent elements, cf. the example, given next. The proof of the statement of Theorem 19 was done in [26] via an application of **Hall´s marriage theorem**. Here, we will present the outline of an alternative proof that is more direct.

EXAMPLE 18:

Figure 12 shows an odd semiorder $\theta = (A, 0)$ of 19 elements by a diagram. The associated minimal schedule length on 2 machines is $t = 14$.

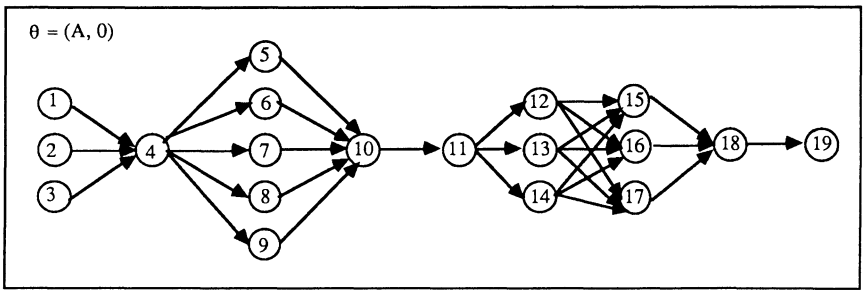


Fig. 12: An odd weak order

THEOREM 19 (cf. [26])
The 2-critical posets are exactly the odd weak orders.

This theorem is e.g. also **implicitly** contained in the correctness proof of certain algorithms for the 2-machine problem [2], [7]. They consider partitions of A into pairwise disjoint sets $A_1,...,A_m$ and U such that $(\alpha, \beta) \in 0$ whenever $\alpha \in A_i$, $\beta \in A_j$ with $i < j$. In other words, the subposets $\theta \mid A_1,... \theta \mid A_m$ are series-connected. Obviously, any such partition gives a lower bound $\sum_{i=1}^{m} \lceil |A_i| / 2 \rceil$ on the optimal schedule length. It is shown in these papers that there exists a partition such that the associated lower bound is in fact **equal** to the length of the constructed schedule, thereby obtaining a correctness proof and an (implicit) Min-Max-Theorem.

Deleting from such an optimal partition the set U, all order relations among elements from the same A_i, and one element from each even A_i then gives an odd weak order in the sense of Theorem 19.

The algorithm in [7] constructs such an optimal partition while producing the linear extension. Basically the idea is to have all elements in θ on their lowest possible **level**, and to traverse the order from the top level by level and construct an optimal schedule. If a level is even, it presents no problem, if it is odd, an element from the closest lower level is pulled up according to a choice rule motivated by the "greedy" principles (make a choice that interferes as little as possible with later pull-up options). The pull-up operation is used to define the partition into the A_i (a pull-up occurs if there is (almost) a series connection between adjacent levels).

All these techniques were used with the aim to prove optimality of the algorithm, and not to give a characterization of the 2-critical posets. We present here a simple proof of the characterization without aiming at their algorithmic construction.

Proof of Theorem 19:
It is easily seen that an odd weak order is 2-critical.

In the other direction, we start from a given 2-critical poset $\theta = (A, 0)$. It is easily verified by an inductive argument that θ is an odd weak order, if θ can properly by decomposed linearly into (autonomous) subposets [26]. If we assume that θ cannot be decomposed this way and, furthermore, know that an optimal unit time schedule for θ on 2 machines needs not more than $\lceil |A| / 2 \rceil$ time units, than obviously θ consists of $|A|$ independent jobs and $|A|$ is uneven, i.e. θ is again an odd weak order. We are therefore done, if we can show that these are all cases occuring. I.e. we will start now from a poset $\theta = (A, 0)$, for which an optimal unit time schedule on 2 machines

needs t time units, where $t > \lceil |A| / 2 \rceil$ (which implies, that θ cannot consist of independent elements). We will show that in this case, θ can properly be decomposed linearly into (autonomous) subposets.

To begin with, we know already, that there are relations $(\alpha, \beta) \in 0$ with $\alpha \neq \beta$. It is easily seen that we can chose α such that all successors of α are maximal elements in θ and that the number of successors of α is minimal under the required property. As θ is 2-critical, elimination of the relation (α,β) from (A, 0) leads to a poset, that can be scheduled in t - 1 time units on 2 machines. Let T denote an associated schedule. Obviously, jobs α and β fill a time slot in this schedule. We can therefore transform T into an optimal schedule T´ of the original problem by only changing $T(\beta)$ to t - 1. It is straightforwardly seen, that all γ fulfilling $T(\gamma) > T(\alpha)$ are exactly the elements from $N(\alpha)$ and that the number of such elements is uneven. Also, the time slot $[T'(\alpha), T'(\alpha) + 1)$ for T´ can, w.l.o.g., be assumed to be the latest not completely filled time slot in any optimal unit time schedule for θ.

We will now show that θ properly decomposes linearly into the (autonomous) subposet $\theta^* = (A^*, 0^*)$ with $A^* = A \setminus N(\alpha)$ and the subposet $\theta \mid N(\alpha)$, consisting of an uneven number $2r + 1$ of independent jobs. Here, the length of an optimal unit time schedule for θ^* is s and the equality $s + r + 1 = t$ holds. What we essentially have to proof is that not only job α, but **any** job from A^* is a predecessor of any job in $N(\alpha)$. Assume this not to be the case for a particular job β_o, where w.l.o.g., β_o can be assumed to be maximal in θ^*. Certainly, if we could schedule θ^* on 2 machines optimally in a way, that only job β_o is scheduled in the last time slot [s, s - 1), a 2-machine schedule for θ with schedule length t - 1 could be obtained, a contradiction. Consequently, any such β_o will never occur in the final time slot. Also, eliminating such a job β_o from θ^* would - for the resulting poset - not decrease the optimal schedule length s on 2 machines. This means, that because of induction, we can find within θ^* a chain $\theta^* \mid L_1 \leq \theta^* \mid L_2 \leq \ldots \leq \theta^* \mid L_k$ of trivial posets $\theta_j^* := \theta \mid L_j$, $j = 1,\ldots,k$, such that θ_j^* consists of $i_j := 2n_j + 1$ independent elements, $j = 1,\ldots,k$ and such that $\sum_{j=1}^{k} (n_j + 1) = s$ holds.

By what was said it is easily obtained that θ^* can be scheduled in such a way, that only the $2n_k + 1$ independent elements from θ_k are scheduled in the time interval $[\sum_{j=1}^{k-1} (n_j + 1), s)$. This implies, that all elements in L_k have to proceed all elements from $N(\alpha)$. Consequently, $\theta_1^* \leq \ldots \leq \theta_k^* \leq \theta \mid N(\alpha)$ forms an odd semiorder of associated minimal unit time schedule length $\sum_{j=1}^{k} (n_j+1) + (r + 1) = t$, lying properly within θ, given that β_o does not belong to the set $\bigcup_{j=1}^{k} L_j$. This is a contradiction to the assumption of θ being 2-critical, concluding the proof. ■

As was already mentioned in [26], the characterization obtained in Theorem 19 yields another way to solve the 2-machine unit time scheduling problem, cf. Section 3. I.e. w.r.t. the simple bounding procedure, discussed in Section 2, one has only to verify, that there will be a bound contradiction, whenever an odd semiorder is considered for a schedule length t', which is too small to find a 2-machine schedule. But this is straightforwardly verified. Thus Theorems 17 and 19 together mean, that the 2-machine bounding procedure indeed works correct for **all** posets.

Some hints to **m-critical** posets for m > 2 are also given in [26]. Again, any such poset can be composed in a (may be trivial) linear sequence of m-critical posets. However, contrary to the 2-machine case, a not further linearly decomposable m-critical poset $\theta = (A, 0)$ may have an optimal m-machine unit time schedule length which properly exceeds $\lceil |A| / m \rceil$, cf. Examples 1 and 13 in this paper. In [26], the hope was expressed that such posets may always be of the type occuring in Examples 1 and 13, i.e. may consist of at most m - 1 independent chains, consisting each of a non-trivial linear sequence of blocks of independent elements. Such posets seem to be very typical as nasty candidates for m-machine scheduling, as potentially there may be problems in filling all m time slots per time unit appropriately. Also, the hope was raised in [26] that for posets of this special type, the correctness of bounding systems as discussed in Section 4 might be proved.

As concluding subject of this paper, we will add some more information to the points just mentioned. We will deal with the "easiest" case available, i.e. 3-machine scheduling of 3-critical posets $\theta = (A, 0)$, that are not linearly decomposable, have at least 3 minimal and 3 maximal elements and that have an optimal unit time schedule length on 3 machines of $t = \lceil |A| / 3 \rceil + 1$. This is exactly the class of posets, covering e.g. Examples 1 and 13 of this paper. We will particularly be concerned with the question of how to obtain new examples of this type by **adding appropriate new maximal elements** to an already found example. In doing so, we want to demonstrate simultaneously the modification of the bounding system approach used in Section 4 for covering all those extensions. Furthermore, we want to obtain more insight into the nature of such 3-critical posets. Actually, we will find examples that are somewhat (but probably not decisively) more complicated as the type suggested above.

EXAMPLE 20:

Consider again the poset $\theta = (A, 0)$, discussed in Example 1 and given by a diagram in Figure 1. This poset has 4-minimal and 7 maximal elements, is not linearly decomposable into autonomous subposets, is 3-critical and needs 5 time units for execution, while $|A| = 12$. Apparently, θ falls into the class of problems considered here. As was already discussed in connection with Example 1, optimal on-line scheduling of this poset will either lead to the bottleneck $\{1,8\}$ at time 2 or to the

bottleneck {i,8} with $i \in \{5, 6, 7\}$ at time 3. If we try to schedule the poset in 4 time units, the bounding system approach would lead to a contradiction. This can already be obtained from the fact that L(8) = 1, U ({1,8}) = 1 and thus U(1) = 1, and U({i,8}) = 2 for $i \in \{5,6,7\}$ and thus U(i) = 2 for $i \in \{5,6,7\}$. This implies $L(8) \geq 2$ and contradicts the bound $U(2) \leq 2$.

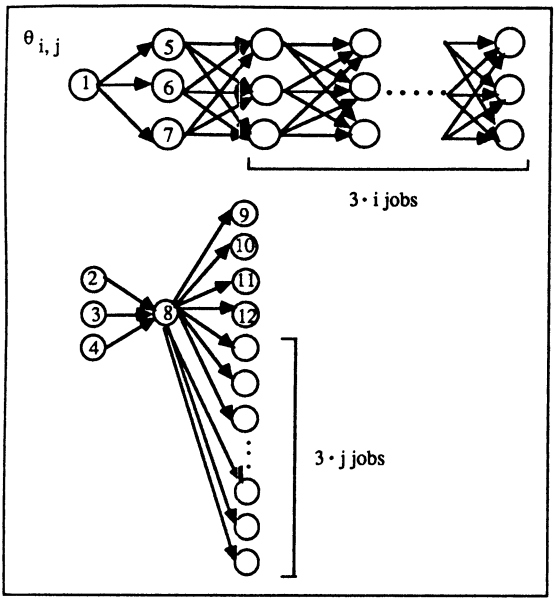


Fig. 13: A class of 3-critical posets θ_{ij}

If we now try to extend θ by adding iteratively new maximal elements, then these elements should be successors of either jobs 1 and 8 and of either jobs $i \in \{5,6,7\}$ and 8. This is needed to allow a reduction of the schedule length, if a relation of type (1,i) with $i \in \{5,6,7\}$ or of type (8,j) with $j \in \{9,...,12\}$ is eliminated. As similar reductions have iteratively to be guaranteed w.r.t. newly introduced elements, we arrive at extensions of the type given in Figure 13 by a diagram and varying parameters $i, j \in \mathbb{N}$.

Obviously, θ_{ij} can optimally be scheduled on 3 machines in 5 + (i + j) time units while the number of jobs is 12 + 3 (i + j). A bound contradiction for the bounding system considered in Section 4 is obtained as follows. Again we have U ({1,8}) = 1 and L(8) = 1, implying U(1) = 1. Also, because of $U(\{i,8\}) = 2$ for $i \in \{5,6,7\}$, U(5) = U(6) = U(7) = 2 will be obtained. If we now proceed this way with the j newly introduced immediate successors of {5,6,7}, these will have to be finished by time 3 and so forth, leading to the completion of the i-th newly introduced 3 jobs at time i + 2. This implies $L(8) \geq i + 3$, while also U(8) = i + 3 is obtained. This is a bound contradiction.

EXAMPLE 21:

We consider the 3-critical poset $\theta = (A, 0)$, given in Figure 14 a by its diagram. The poset is of the type, wanted, and was obtained as a modification of the poset given in Figure 1 and used in Example 20. The example leads to a class of even more involved 3-critical posets, as considered here.

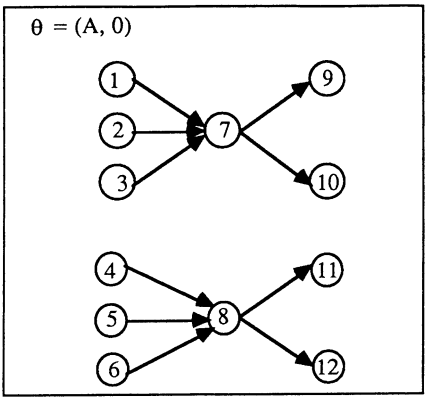


Fig. 14a: A 3-critical poset

The only possible extension by 3 elements of $\theta = (A, 0)$, that are maximal in the resulting poset $\theta^* = (A^*, 0^*)$, which is 3-critical of the type considered, is given in Figure 14b by a diagram.

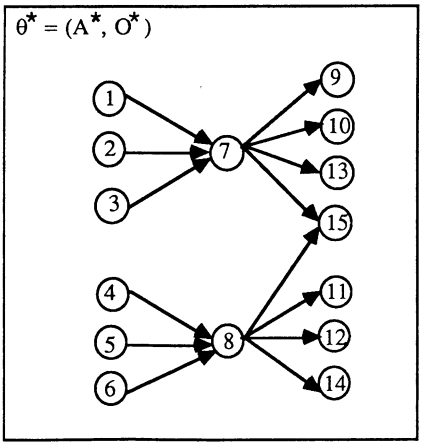


Fig. 14b: A 3-critical poset, that is an extension of the 3-critical poset in Fig. 14a

(1) Note that here we obtained an example that consists **no longer of two independent chains** of linear sequences of independent blocks, as job 15 follows jobs 7 and 8. Still, the deviation from this expected type is not too far, given that job 15 is maximal. In fact, another 3-critical poset with this phenomena is obtained from $\theta = (A, 0)$ in Figure 9 in Example 14, when job 21 is made an immediate successor of jobs $\{1,...,6\}$.

(2) An interesting property of θ^* is the fact that it is not possible to take away 3 minimal elements from θ^* and still remain 3-critical. Luckily, the elimination of e.g. jobs $\{13,14,15\}$ preserves the property of being 3-critical. For induction proofs it would be very advantageous to establish the result, that for the problem instances considered here, either 3 minimal or 3 maximal elements can be taken away, such that the remaining poset is still 3-critical, except for the case that already a chain of blocks consisting of independent elements give the reduced schedule length. This is what will happen in the poset used in Example 13, regardless of how we take away 3 minimal or maximal elements.

(3) Further 3-critical extensions θ^*_{ijk} with $i, j, k \in \mathbb{N}$ of θ^* as considered are obtained by adding a chain of i blocks of 3 elements each, following jobs $\{9,10,13\}$, a similar chain of j blocks of 3 elements each, following jobs $\{11,12,14\}$ and further 3k independent elements, following jobs 7 and 8 (independent from job 15). These posets have $15 + 3(i + j + k)$ elements and the associated optimal 3-machine unit time schedule length is $6 + (i + j + k)$. A bound contradiction for all these examples w.r.t. the schedule length $5 + i + j + k$ can be obtained completely analogously to the argumentation in Example 20.

6. CONCLUDING REMARKS:

The authors hope that this report on ongoing research will further stimulate work on a solution of the general m-machine unit time scheduling problem. The insights presented and the examples given might prove helpful in this respect. From the point of view of the authors, concentration on a characterization of m-critical posets, e.g. for $m = 3$, might be particularly worthwhile at the moment, also concerning a proof of membership to CO-NP, but even more for finding polynomial algorithms. There were given strong indications w.r.t. the special nature of such posets. Also, these structures really seem to help in proofs on the correctness of the proposed bounding systems, particularly again w.r.t. the 3-machine case. As was shown in the paper, a general answer for these steps would yield the general polynomial time solution procedure wanted.

Presently, the two last authors are further looking into these questions, together with different co-workers, in particular also students from the University of Passau. In this context, special classes of posets are studied in more detail and the algorithmic procedures mentioned in this paper

are all encoded and tested in typical examples. Progress in all these aspects will be made available on request and will possibly also be published later at some appropriate point of time.

ACKNOWLEDGEMENT

We want to thank J.K. Lenstra (Amsterdam) and A.H.G. Rinnooy Kan (Rotterdam), who, during a conference in Durham (U.K.) brought the m-machine unit time scheduling problem - and its significance - to our attention. Also we want to mention here the implementation work of Mr. G. Lutzeier, Passau, who testet all the algorithms, newly introduced in this paper.

REFERENCES:

[1] Bartusch M., Möhring R.H., Radermacher F.J.: Some remarks on the m-machine unit time scheduling problem, in: Noltemeier H., Proceedings of the WG '85, Trauner Verlag, Linz, p. 23-26, 1985

[2] Coffman E.G., Graham R.L.: Optimal scheduling for two processor systems, Acta Informatica 1, p. 200-213, 1972

[3] Dolev D., Warmuth M.K.: Scheduling Precedence Graphs of Bounded Height, Journal of Algorithms 5, p. 48-59, 1984

[4] Dolev D., Warmuth M.K.: Scheduling Flat Graphs, Research Report RJ 3398, IBM, San Jose, CA, 1984

[5] Fishburn P.C., Intransitive indifference in preference theory: A survey, Oper. Res. 18 p. 207-228, 1970

[6] Fujii M., Kasami T., Ninomiyak K.: Optimal sequencing of two equivalent processors, SIAM J. Appl. Math. 17, 1969, Erratum 20, p. 141, p. 784-789, 1971

[7] Gabow H.N.: An almost-linear algorithm for two-processor scheduling, J. ACM 29, p. 766-780, 1982

[8] Garey M.R., Johnson D.S.: Scheduling tasks with nonuniform deadlines on two processors, J. Assoc. Comp. Mach. 23, p. 461-467, 1976

[9] Garey M.R., Johnson D.S.: Two-Processor Scheduling with Start-Times and deadlines, SIAM J. Comput., 6, p. 416-426, 1977

[10] Garey M.R., Johnson D.S.: Computers and intractability: A guide to the theory of NP-completeness, Freeman, San Francisco 1979

[11] Gilmore P.C., Hoffman A.J.: A characterization of comparability graphs and of interval graphs, Canad. J. Math. 16, p. 549-548, 1964

[12] Golumbic M.C., Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980

[13] Hsu N.C.: Elementary proof of Hu's theorem on isoton mappings, Proc. Amer. Math. Soc. 17, p. 111-114, 1966

[14] Hu T.C.: Parallel Sequencing and Assembly Line Problems, Operations Research, 9, pp. 841-848, 1961

[15] Igelmund G., Radermacher F.J.: Preselective strategies for the optimization of stochastic project networks under resource contraints, in: Networks, 13, p. 1-29, 1982

[16] Igelmund G., Radermacher F.J.: Algorithmic approaches to preselective strategies for stochastic scheduling problems, in: Networks, 13, p. 29-48, 1983

[17] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.: Recent developments in deterministic sequencing and scheduling: A survey in: Dempster M.A.H. et al. (Hrsg.) Deterministic and Stochastic Scheduling, Reidel, Dordrecht, pp. 35-73, 1982

[18] Lenstra J.K., Rinnooy Kan A.H.G.: Complexity of Scheduling Under Precedence Constraints, Operations Research, 26, pp. 22-35, 1978

[19] Möhring R.H.: Algorithmic aspects of comparability graphs and interval graphs, in: Rival I. (Hrsg.): Graphs and Order, pp. 41-101, Reidel, Dordrecht 1984/85

[20] Möhring R.H.: On the computational complexity of the substitution decomposition for relations, set systems and Boolean functions, in: Annals of Operations Research, p. 195-225, 1985/86

[21] Möhring R.H., Radermacher F.J.: Substitution decomposition of discrete structures and connections with combinatorial optimization, in: Ann. Discrete Math., 19, p. 257-356, 1984

[22] Papadimitrious, C.H., Yannakakis M.: Scheduling Interval-Ordered Tasks, SIAM J. Comput., 8, 1979

[23] Pelzer H.: Zur Komplexität des 3-Maşchinen-Schedulingproblems, diploma thesis, RWTH Aachen, 1983

[24] Radermacher F.J.: Scheduling of project networks, in: Annals of Operations Research 4, p. 227-252, 1985/86

[25] Radermacher F.J.: Schedule-induced posets, Discrete Applied Mathematics 14, North Holland, p. 67-91, 1986

[26] Radermacher F.J.: Characterization of the critical posets in two-machine unit time scheduling; appears in: Opitz O., Rauhut B. (eds.): Ökonomie und Mathematik, Springer Verlag, Heidelberg, 1987

[27] Sethi R.: Algorithms for minimal length schedule, in: Coffman E.G. (ed.): Computer and Job-shop scheduling theory, Wiley, New York, p. 51-99, 1976

[28] Ullmann J.D.: NP-complete Scheduling Problems, Journal of Computer and System Science, 10, pp. 384-393, 1975

ON INVERSE-IMAGE OF NON-ORIENTED GRAPHS

S. Perz and S. Rolewicz (Warszawa)

By a non-oriented graph we shall call a pair $G=\langle V,A\rangle$, where V is a finite set, and A is a family of two-elements subsets of V. The elements of V we shall call vertices, the elements of A we shall call edges.
In sequel we shall consider only non-oriented graphs and we shall call them briefly graphs.

Let U be a subset of V, $U \subset V$. We write $A|_U$ a family consisting of two-elements subsets of U belonging to A.

Of course the pair

$$G|_U=\langle U,A|_U\rangle$$

is also a graph. We shall call the graph $G|_U$ a subgraph of the graph G.
Let a graph $G=\langle V,A\rangle$ be given. A subgraph $G_o=\langle V_o,A_o\rangle$ is called a clique if G_o is a complete graph, in other words A_o is a set of all two-elements subsets of V_o. A subgraph $G_1=\langle V_1,A_1\rangle$ is called an anticlique (or stable set) if G_1 is an isolated graph, in other word A_1 is empty.

Let $G=\langle V,A\rangle$ be a graph. By a complement of the graph G we shall call the graph $\bar{G}=\langle V,\bar{A}\rangle$, where $\bar{A}$ consist of those two-elements sets which does not belong to A.
By a chromatic number $\chi(G)$ of a graph $G=\langle V,A\rangle$ we shall call the minimal number of anticliques covering V. The maximal number of vertices in a clique contained in G we shall call the clique number and we shall denote it $\omega(G)$.

A graph $G=\langle V,A\rangle$ is called perfect if

$$\chi(G\big|_U)=\omega(G\big|_U) \tag{1}$$

for all $U\subset V$.

Now we consider two graphs $G=\langle V,A\rangle$ and $G^{-1}=\langle W,D\rangle$.
Let f be a function mapping W onto V. The graph G^{-1} we shall call an inverse image of the graph G if for arbitrary $v_1,v_2\in V$ such that $\{v_1,v_2\}\in A$ for arbitrary w_1,w_2 such that $f(w_1)=u_1$, $f_2(w_2)=u_2$, $\{w_1,w_2\}\in D$.

Of course for each $v\in V$ $f^{-1}(v)$ is a subset of W. Thus we can consider a subgraph $G^{-1}\big|_{f^{-1}(v)}=\langle f^{-1}(v),\ D\big|_{f^{-1}(v)}\rangle$. Since V is finite we can numerate elements v_n writing $v_1,\ldots,v_n$. We shall briefly write

$$G^{-1}\big|_{f^{-1}(v_i)}=G_i, f^{-1}(v_i)=V_i,$$

$$D\big|_{f^{-1}(v_i)}=A_i.$$

Theorem 1. A graph $G^{-1}=f^{-1}(G)$ is perfect if and only if the graphs $G=\langle V,A\rangle$, $G_i=\langle V_i,A_i\rangle$, $i=1,2,\ldots,n$ are perfect.
The special extremal cases of Theorem 1 are well known. Namely Theorem 1 are known when G_i $i=1,\ldots,n$ full graph (see [9], [10]) and when G_i, $i=1,2,\ldots,n$ are isolated graphs (see for example [6]).

The authors do not know any paper containing the proof of Theorem 1. However they estimate that the problem of priority of Theorem 1 is not essential for them. Their idea is to present another approach to the graph theory based on functional analysis, and a notion of norms.
This approach was introduced by the first author [15]. It is a contimation and developing of approach to the graph theory via convex polyhedra introduced by Fulkerson ([4], [5]) and Padberg

([11]-[14]). The first essential observation in this approach is that the problem of perfectness of the graph is much better fitted in the framework of the hypergraph theory than in the graph theory itself.

We recall that a hypergraph is a pair $G=\langle V,C\rangle$, when C is an arbitrary family of subsets of V. Let a graph $G=\langle V,A\rangle$ be given. By A_c we denote the family of all subsets $U\subset V$ such that $G|_U$ is a clique. Thus $G_c=\langle V,A_c\rangle$ is a hypergraph corresponding to the graph G. The family A_c has the following properties

(a) each one-element set belongs to A_c

(b) if $a\in A_c$ and $b\subset a$ then $b\in A_c$

(c) if all two-elements subsets of a set a belong to A_c, then $a\in A_c$.

It is easy to observe that each family satisfying (a)-(c) induces a graph $G=\langle V,A\rangle$ where A is defined as all two-elements set of A_c. Therefore there is a one-to-one correspondance between the graph $G=\langle V,A\rangle$ and the hypergraph $G_c=\langle V,A_c\rangle$.

Let $G=\langle V,A\rangle$ be a graph let $\bar{G}=\langle V,\bar{A}\rangle$ be its complement. The both graphs induces the corresponding hypergraphs

$$G_c=\langle V,A_c\rangle \quad \text{and} \quad \bar{G}_c=\langle V,\bar{A}_c\rangle.$$

It is easy to verify that $\bar{A}_c$ consist of all anticliques of A and such called transversality condition holds i.e. the cardinality of a common part of a and b is at most one:

$$(1) \qquad \#a\cap b\leq 1$$

for all $a\in A_c, b\in\bar{A}_c$.

The set of all b satisfying (1) for all $a\in A$ we shall call the transversal set to A and denote by $(A)^{tr}$. Of course

$$(2) \qquad A\subset((A)^{tr})^{tr}.$$

If the equality holds i.e. $A=((A)^{tr})^{tr}$ we shall call that the hypergraph $G=\langle V,A\rangle$ is conformal. It is easy to observe that a hypergraph is conformal if and only if (a)-(c) holds i.e. if it is generated by a certain graph.

Let certain conformal hypergraph $G_c=\langle V,A_c\rangle$ be given. Let n be equal the cardinality of V, $\#V=n$. We shall order to the hypergraph G_c certain subset $\mathcal{A}$ of R^n. Namely we enumerate all vertices (elements of V). Thus elements of $\mathbb{R}^n$ can be considered as a real valued functions defined on V. The characteristic function of a set $a\subset V$ is a function

$$\chi_a(v_i) = \begin{cases} 1 \text{ for } v_i\in a \\ 0 \text{ for } v_i\notin a. \end{cases}$$

By H_a we denote the set of $2^{\#a}$ functions consisting of those function α that $|\alpha(v_i)|=\chi_a(v_i)$. The set $\mathcal{A}=\bigcup_{a\in A}H_a$. The set $\mathcal{A}$ induces on the space $\mathcal{X}^n$ consisting of vector of integer valued coefficients so called König norm in the honour of the fundamental work of König (see [5], [6])

$$\|x\|_A=\inf\{\sum_{i=1}^n|\alpha_i|:x=\sum_{i=1}^n\alpha_i a_i,\ a_i\in\mathcal{A},\alpha_i\text{-integers}\}$$

and so called Minkowski norm

$$|x|_A=\inf\{\sum_{i=1}^n|\alpha_i|:x=\sum_{i=1}^n\alpha_i a_i,\ a_i\in\mathcal{A},\alpha_i\text{-reals}\}.$$

Of course the Minkowski norm is well determined on the whole R^n and can be also defined there by the classical formula

$$|x|_A=\inf\{t>0:\frac{x}{t}\in\operatorname{conv}\mathcal{A}\}.$$

Observe that the both norms satisfies the triangle inequality

$$\|x+y\|_A \leq \|x\|_A + \|y\|_A$$

$$|x+y|_A \leq |x|_A + |y|_A.$$

If $x=0$ then $\|x\|_A=0=|x|_A$ and conversely if $\|x\|_A=0$ (or $|x|_A=0$) then $x=0$.
Moreover if $|x|=(|x_1|,\ldots|x_n|)$ then

$$\| |x| \|_A = \|x\|_A, \quad \| |x| |_A = |x|_A.$$

Observe that $0 \leq |x|_A \leq \|x\|_A$. The Minkowski norm is homogeneous $\|tx\|_A=|t|\,|x|_A$.

Let a graph $G=\langle V,A\rangle$ be given. By $G_c=\langle V,A_c\rangle$ we denote the corresponding hypergraph, constructed in the way described previously. Let $U \subset V$ be an arbitrary subset of V and let $m=\chi_U$ denote the characteristic function of the set U. Observe that the clique number $\omega(G|_U)$ of the graph $G|_U$ is equal to

$$\omega(G|_U)=\max_{a\in A_c} \#m\cap a$$

$$=\max_{a\in \mathcal{A}} (m,a)=\max_{a\in \operatorname{conv}\mathcal{A}} (m,a)=|m|^A$$

where (m,a) denotes the inner product and $|m|^A$ denotes the norm conjugate to the Minkowski norm $|x|^A$.

Let $\bar{G}=\langle V,\bar{A}\rangle$ denote the complement of the graph G and let $\bar{G}_c=\langle V,\bar{A}_c\rangle$ denote the hypergraph corresponding to G_c.

Let $U \subset V$ and let $m=\chi_U$ be a characteristic function of U. It is easy to observe that the chromatic number $\chi(G|_U)$ equal to the minimal covering of U by anticliques i.e. $\chi(G|_U)=\|m\|_{\bar{A}}$, where $\|m\|_{\bar{A}}$ denotes the König norm in $\bar{G}_c=\langle V,\bar{A}_c\rangle$ induced by $\bar{A}_c$.

Thus a graph $G=\langle V,A\rangle$ is perfect if and only if for any characteristic function $m=\chi_U$ we have

$$||m||_{\bar{A}} = |m|^{A}.$$

This equality leads to a series of equivalent conditions, proved by the first author ([15]).

Theorem 2. Let $G=\langle V,A\rangle$ be a given graph. Then the following conditions are equivalent

(a) the graph G is perfect

(b) the graph $\bar{G}$ is perfect

(c) $||m||_{\bar{A}} = |m|^{A}$

(d) $||m||_{A} = |m|^{\bar{A}}$

(e) $||x||_{\bar{A}} = |x|^{A}$ for all $x \in Z^n$

(f) $||x||_{A} = |x|^{\bar{A}}$ for all $x \in Z^n$

(g) the König norm $||x||_{\bar{A}}$ is homogeneous for $k \in Z$

(h) the König norm $||x||_{A}$ is homogeneous for $k \in Z$

(i) $\lim \frac{1}{k} ||kx||_{\bar{A}}$ is an integer

(j) $\lim \frac{1}{k} ||kx||_{A}$ is an integer

(k) the sets conv A, conv $\bar{A}$ are polar each to the second in $\mathbb{R}^n$.

As a consequence of Theorem 2 we obtain

Corollary 1 (see [1]). A graph $G=\langle V,A\rangle$ is perfect if and only if the polar set to the set conv $\mathcal{a}$ has only extremal point of the form $\{\varepsilon_i, \ldots \varepsilon_n\}$ where ε_i is equal either to 0 or to 1 or to -1.

Proof. If the polar set to the set conv $\mathcal{a}$ is conv $\bar{\mathcal{a}}$ then its extremal points belong to $\bar{\mathcal{a}}$. Thus they are of requested form. Conversely if the point m of the requested form belongs to the polar set to conv $\mathcal{a}$ then $\sup_{x \in \mathcal{a}} (m,x) \leq 1$. Thus the support of m belongs to $\bar{A}$ and $m \in \bar{\mathcal{a}}$.

The characterisation of the perfect graph given by Corollary 1 will be used in our proof of Theorem 1.

Before starting the proof we shall recall a classical construction of product space with respect to a given space done by Day

[2] (see [3]).

Let $(X, \|\ \|)$ be a Banach space of dimension n. The space X is called solid if for $x=(x_1,\ldots x_n)$ $y=(y_1,\ldots y_n)$ the condition $|y_i| \leq |x_i|$ $i=1,2,\ldots n$ implies that $\|y\| \leq \|x\|$. By the construction of the set $\mathcal{A}$ it is easy to see that the space $(\mathbb{R}^n, |\ |_A)$ is a solid space.

Let $(X, \|\ \|)$ be a solid Banach space. Let $(X_i, \|\ \|_i)$ $i=1,\ldots,n$ be a system of n Banach spaces. We construct a new space called the product of spaces X_i in the sense of X in the following way. We take the carthesian product $X_1 \times \ldots \times X_n$ and we equip it in the norm

$$|||(x_1,\ldots,x_n)||| = \|(\|x_1\|_1,\ldots \|x_n\|_n)\| .$$

The space $(X_1 \times \ldots \times X_n, |||\ \ |||)$ we shall denote briefly $(X_i)_X$.

Let $G=\langle V,A\rangle$, $G=\langle V_i,A_i\rangle$ denote the graphs described on the beginning of this paper. We construct a corresponding sets $\mathcal{A}$, $\mathcal{A}_i$, $i=1\ldots n$. And we order to the each of those graph the corresponding Minkowski norm. In this way we obtain n+1 solid Banach spaces $(X, |||\ |||_A)$, $(X_i, |||\ |||_{A_i})$. The dimension of X is n. Now we shall consider the product of the spaces X_i with respect to the space X, $(X_i)_X$ and we shall show that the space $((X_i)_X, |||\ |||)$ is the space generated by the graph $G_1=\langle W,D\rangle$. The proof will be based on the following classical theorem.

Theorem 3. Let $(X, \|\ \|)$ be an n-dimensional solid Banach space. Let $(X_i, \|\ \|_i)$ $i=1,\ldots n$ be finite dimensional. A point $x=(x_1,,,x_n)$ is an extremal point of the unit ball $U_{(X_i)_X}=\{x: |||x||| \leq 1\}$ in the space $((X_i)_X, |||\ |||)$ if and only if

(a) the point $(\|x_1\|_1,\ldots,\|x_n\|_n)$ is an extremal point of the unit ball $U_X=\{x: \|x\| \leq 1\}$ of the space $(X, \|\ \|)$

and

(b) x_i is an extremal point of the ball $B_i(\|x_i\|_i)$ of the radius $\|x_i\|_i$ in the space X_i, $i=1,2,\ldots,n$.

Proof. Let $x=(x_1\ldots x_n)$ be an element of norm one, $|||x|||=1$. If x is not an extremal point of $U_{(X_i)_X}$ then there are two elements $u=(u_1,\ldots u_n)$ and $v=(v_1,\ldots v_n)$ of norm one, $|||u|||=1$, $|||v|||=1$ such that $x=\frac{u+v}{2}$ and $u\neq x\neq v$.

By triangle inequality

$$\|x_i\|_i \leq \frac{1}{2}(\|u_i\|_i + \|v_i\|_i)$$

and since element $(\|x_i\|_i)$ $(\|u_i\|_i)$ and $(\|v_i\|_i)$ has norm one in the space X, then we must have equality

$$\|x_i\|_i = \frac{1}{2}(\|u_i\|_i + \|v_i\|_i).$$

Now there are two possibilities, either

(i) there is i such that $\|x_i\|_i \neq \|u_i\|_i$. In this case (a) does not hold

or

(ii) $\|x_i\|_i = \|u_i\|_i = \|v_i\|_i$.

Since $x\neq u$, in the second case there is an index i_o such that $x_{i_o}\neq u_{i_o}$ and it implies that (b) does not hold.

Conversely suppose now that (a) does not hold i.e. $(\|x_1\|_1,\ldots \|x_n\|_n)$ is not an extremal point of the unit ball in U_X. It means that there are two systems of nonnegative numbers $(a_1\ldots a_n)$, $(b_1,\ldots b_n)$ belonging to the unit ball U_X such that $(a_1,\ldots a_n)\neq(\|x_1\|_1,\ldots,\|x_n\|_n)$ and

$$\|x_i\|_i = \frac{a_i+b_i}{2} \quad i=1,\ldots,n.$$

Let $u_i=a_i\frac{x_i}{\|x_i\|_i}$, $v_i=b_i\frac{x_i}{\|x_i\|_i}$ if $\|x_i\|_i\neq 0$. If $\|x_i\|_i=0$ we put $u_i=$

$v_i=0$. It is easy to observe that $u=(u_1,\ldots,u_n)$, $v=(v_1,\ldots,v_n)$ are different than $x=(x_1,\ldots,x_n)$ the norm $|||u|||=|||v|||=1$ and moreover $x=\frac{u+v}{2}$ i.e. x is not an extremal point of the unit ball $U_{(X_i)_X}$.

Suppose now that (b) does not hold i.e. there is an index i_o such that x_{i_o} is not an extremal point of the ball $B_{i_o}(\|x_{i_o}\|_{i_o})$. Thus there are two elements of x_{i_o}, u_{i_o} and v_{i_o} such that

$$\|x_{i_o}\|_{i_o}=\|u_{i_o}\|_{i_o}=\|v_{i_o}\|_{i_o},$$

$$x_{i_o}=\frac{u_{i_o}+v_{i_o}}{2}$$

and $x_{i_o}\neq u_{i_o}$.

Let $u_i=\begin{cases} x_i & \text{for } i\neq i_o \\ u_{i_o} & \text{for } i=i_o \end{cases}$ $v_i=\begin{cases} x_i & \text{for } i\neq i_o \\ v_{i_o} & \end{cases}$. Thus $x=\frac{u+v}{2}$, $x\neq u$ $|||u|||=|=|||v|||$ and x is not an extremal point of $U_{(x)_{x_i}}$.
As a consequence of Theorem 3 we obtain

Theorem 4. Let $G=\langle V,A\rangle$ be a given graph. Let $G^{-1}=\langle W,D\rangle$ be an inverse image of the graph G. Let $V_i=f^{-1}(v_i)$ and $A_i=D|_{V_i}$. Let $(X,\|\ \|)$ be a solid Banach space corresponding to the graph $G=\langle V,A\rangle$. Let $(X_i,\|\ \|_i)$ be Banach spaces corresponding to the graphs $G_i=\langle V_i,A_i\rangle$. Let $(Y,\|\ \|_Y)$ be a Banach space corresponding to the inverse image $G^{-1}=\langle W,D\rangle$. Then $(Y,\|\ \|_Y)=((X_i)_x,|||\ \ |||)$.

Proof. Let d be an arbitrary element of D_i. Let $m=(m_1,\ldots,m_n)$ be an arbitrary element of H_d i.e. such an element that $|m|=(|m_1|,\ldots |m_n|)$ is a characteristic function of d. By the definitions $m\in\mathcal{D}$. Since $d\cap X_i\in A_{i,c}$ we have that $x_i=m|_{A_i}$ belongs to $\mathcal{A}_i$ and $\|x_i\|=1$. Since by the definition of the inverse image $(\|x_1\|,\ldots\|x_n\|)^i\in\mathcal{A}$ we obtain that $|||m|||\leq 1$. On the other hand let $x=(x_1\ldots x_n)^n$ be an extremal point of the unit ball $U_{(X_i)_X}$.

Now we shall use Theorem 3. By property (a) $(\|x_1\|_1,\ldots\|x_n\|_n)$ is an extremal point of the unit ball of the space X. Thus $\|x_i\|_i$ is equal either 0 other 1 and the support of $(\|x_1\|_1,\ldots \|x_n\|_n)$ belongs to A. By (b) x_i is an extremal point of the unit ball of the space X_i. Therefore $x_i=(x_{i,j})$ where $x_{i,j}$ is either equal 0 or 1 or -1 and the support of x_i belongs to A_i. Those conditions and the definition of D and $\mathcal{D}$ implies that $x\in\mathcal{D}$. Since the unit ball in the finite dimensional space is uniquely determined by its extremal point Theorem 4 holds.

For the proof of Theorem 1 we need a following classical

Theorem 5. Let $(X,\|\ \|)$, $(X_i,\|\ \|_i)$ $i=1,2\ldots n$ be finite-dimensional Banach spaces. Let dim X=n and let X be a solid space. Let $(X,|||\ \ |||)$ be the product of the spaces $(X,\|\ \|_i)$ $i=1,2\ldots n$ in the sense of space $(X,\|\ \|)$. Then the conjugate space has norm given by formula

$$|||f|||^*=\|(\|f_1\|_1^*,\ldots\|f_n\|_n^*)\|^* \tag{5}$$

where $f=(f_1\ldots f_n)$, $f_i\in(x_i^*,\|\ \|_i^*)$ and the norms $\|\ \|^*,\|\ \|_i^*$ $i=1,2\ldots n$ denotes the norms conjugate to the norms $\|\ \|,\|\ \|_i$ $i=1,2,\ldots,n$.

Proof. Observe that for any $f\in Y^*$ and $x\in Y$

$$|f(x)|\leq\sum_{i=1}^n|f(x_i)|=\sum_{i=1}^n|f_i(x_i)|$$

$$\leq\sum_{i=1}^n\|f_i\|_i^*\|x_i\|_i\leq$$

$$\leq\|(\|f_1\|_1^*,\ldots\|f_n\|_n^*)\|^*$$

$$\|(\|x_1\|_1,\ldots,\|x_n\|_n)\|\leq|||f|||^*\ \ |||x|||_*$$

what implies that the norm of the functional f is not greater than $|||f|||^*$. On the other hand let $f=(f_1,\ldots f_n)$ be an arbitrary element of the conjugate space. By the definition of $|||f|||^*$ there is a

system of nonnegative numbers $(a_1,a_2\ldots a_n)$ such that $\|(a_1,\ldots, a_n)\|=1$ and $|||f|||^*=\sum_{i=1}^n \|f_i\|_i a_i$.

Now for each i we shall take x_i such that $\|x_i\|_i=a_i$ and $f_i(x_i)=\|f_i\|_i a_i$.
Thus

$$|||f|||^* =\sum_{i=1}^n f_i(x_i)=f(x)$$

and this shows that $|||f|||^*$ is attained.

Proof of Theorem 1. Let X denote the solid Banach space generated by the graph $G=\langle V,A\rangle$. Let X_i denote the Banach space generated by the graphs $G_i=\langle V_i,A_i\rangle$. Since those graphs are perfect, by Corollary 1 the unit balls of the conjugate spaces X^*, X_i^* i=1,2...n has only points of the form $\{\varepsilon_i\}$ where ε_i is equal either zero or 1 or -1.
Thus by Theorem 4 the extremal points of the ball in the product of X_i with respect to X are of this same form. Thus by Corollary 1 the graph $G^{-1}=\langle W,D\rangle$ is perfect.

References

[1] V. Chvatal, On certain polytopes associated with graphs, Jour.Combin.Theory B 18 (1975), 138-154.

[2] M.M.Day, Reflexive spaces not isomorphic to uniformly convex spaces, Bull.Amer.Math.Soc. 47 (1941), 313-317.

[3] M.M.Day, Normed linear spaces, Springer Verlag 1958.

[4] D.R.Fulkerson, Blocking and antiblocking pair of polyhedra, Math.Programming 1 (1971), 168-194.

[5] D.R.Fulkerson, Anti-blocking polyhedra, Jour.Combin.Theory 12 (1972), 50-71.

[6] M.C.Golumbic, Algorithmic graph theory and perfect graphs, Academic Press 1986.

[7] D.König, Über eine Schlußweise aus dem Endlichen und Unendlichen, Acta, Litt.Acad.Sc. Univ.Hung Franc Josephinae, Sectio Sc.Math. 3 (1927), 121-130.

[8] D.König, Theorie der endlichen und unendlichen Graphen, Leipzig 1936.

[9] L.Lovász, Normal hypergraphs and the perfect graph conjecture, Discrete Math. 2 (1972), 253-267.

[10] L.Lovász, A characterization of perfect graph, J.Combin. Theory Ser. B 13 (1972), 95-98.

[11] M.Padberg, On the final structure of set packing polyhedra, Math.Programming 5 (1973), 199-215.

[12] M.Padberg, Perfect zero-one matrices, Math.Programming 6 (1974), 199-215.

[13] M.Padberg, A note on the total unimodularity of matrices, Discrete Math. 14 (1976), 273-278.

[14] M.Padberg, Almost integral polyhedra related to certain optimization problem, Linear Algebra and Appl. 15 (1976), 69-88.

[15] S.Perz, Graphs and norms (in Polish), Ph.D.dissertation, Institute of Mathematics of the Polish Academy of Sciences

IV. Differential Equations and Operator Theory

Existence of Optimal Trajectory of Mayer Problem for Neutral Functional Differential Inclusions

Michał Kisielewicz
Zielona Góra, Poland

Introduction. Existence theorems for optimal control problems for systems described by neutral functional-differential inclusions can be obtained as a consequence of the compactness of the set of attainable trajectories and lower or upper semi-continuities given functionals. Therefore, we begin our lecture with some sufficient conditions for the compactness of the set $T(D,F,A,B)$ of all absolutely continuous functions $x:[\sigma-r,\sigma+a] \to \mathbb{R}^n$ satisfying the following conditions

$$(1) \qquad \begin{cases} \dot{x}(t) \in F(t,x_t,\dot{x}_t) \text{ for a. e. } t\in[\sigma,\sigma+a] \\ (t,x_t)\in A \text{ for } t\in[\sigma,\sigma+a] \\ (\sigma,x_\sigma,\sigma+a,x_{\sigma+a})\in B, \end{cases}$$

where D,A,B are nonempty subsets of $R\times C([-r,0],\mathbb{R}^n)\times L([-r,0],\mathbb{R}^n)$, $R\times AC([-r,0],\mathbb{R}^n)$ and $R\times C([-r,0],\mathbb{R}^n)\times R\times C([-r,0],\mathbb{R}^n)$, respectively and F is a set-valued function defined on D with values in the space $Conv(\mathbb{R}^n)$ of all nonempty compact and convex subsets of $\mathbb{R}^n$. Here x_t and $\dot{x}_t$ denote for given $x:[\sigma-r,\sigma+a]\to\mathbb{R}^n$ and a fixed $t\in[\sigma,\sigma+a]$ mappings defined on $[-r,0]$ by setting $x_t(s)=x(t+s)$ and $\dot{x}_t(s)=\dot{x}(t+s)$ for $s\in[-r,0]$.

In what follows we shall assume that sets D,A and B are such that pairs (D,A) and (A,B) have the following properties

(i) (D,A) is conformable, i.e., for every (t,x), $(t,y)\in A$ we have $(t,x,\dot{y})\in D$;

(ii) (A,B) has a nonempty intersection property, i.e., it is such that a set $K(A,B)\triangleq\{(x;[\sigma-r,\sigma+a])\in \mathcal{K}_r(A):(t,x_t)\in A$ for $t\in[\sigma,\sigma+a],(\sigma,x_\sigma,\sigma+a,x_{\sigma+a})\in B\}$ is nonempty.

Here and latter, $\mathcal{K}_r(A)\triangleq \bigcup_{\alpha\le\sigma<\beta}\ \bigcup_{0<a<\beta-\sigma} AC([\sigma-r,\sigma+a],\mathbb{R}^n)$, where A is assumed to have a connected projection on the real line, say equal to an interval $[\alpha,\beta]$ such that $\beta-\alpha>r\ge 0$. As usual (see [1]) $\mathcal{K}_r(A)$ is considered as a metric space with a metric d defined by

$$(2)\qquad d[(x_1;[\sigma_1-r,\sigma_1+a_1]),(x_2;[\sigma_2-r,\sigma_2+a_2])]\triangleq \max[|\sigma_1-\sigma_2|,|a_1-a_2|,\|x_1^A-x_2^A\|_{\alpha\beta}],$$

where for $x_i\in AC([\sigma_i-r,\sigma_i+a_i],\mathbb{R}^n)$, $i=1,2$, we define $x_i^A(t)=x_i(\sigma_i-r)$ for $t\in[\alpha-r,\sigma_i-r)$, $x_i^A(t)=x_i(t)$ for $t\in[\sigma_i-r,\sigma_i+a_i]$, $x_i^A(t)=x_i(\sigma_i+a_i)$ for $t\in(\sigma_i+a_i,\beta]$ and $\|x_i^A\|_{\alpha\beta}\triangleq|x_i^A(\alpha-r)|+\int_\alpha^\beta|\dot{x}_i^A(t)|\,dt$.
In a similar way we define a metric space $(C_r(A),\rho)$ with $C_r(A)=\bigcup_{\alpha\le\sigma<\beta}\ \bigcup_{0<a<\beta-\sigma} C([\sigma-r,\sigma+a],\mathbb{R}^n)$ and ρ defined by

$$(3)\qquad \rho[(x_1;[\sigma_1-r,\sigma_1+a_1]),(x_2;[\sigma_2-r,\sigma_2+a_2])]\triangleq \max[|\sigma_1-\sigma_2|,|a_1-a_2|,|x_1-x_2|_{\alpha\beta}],$$

where $|\cdot|_{\alpha\beta}$ denotes the supremum norm of $C([\alpha-r,\beta],\mathbb{R}^n)$.

Given a sequence $\{(x_n;[\sigma_n-r,\sigma_n+a_n])\}$ of $\mathcal{K}_r(A)$ is called d-weakly converging to $(x;[\sigma-r,\sigma+a])$ if $\sigma_n\to\sigma$, $a_n\to a$ and $x_n^A\rightharpoonup x^A$ as $n\to\infty$, where $\rightharpoonup$ denotes the weak convergence in $AC([\alpha-r,\beta],\mathbb{R}^n)$. We will say that a set $S\subset\mathcal{K}_r(A)\subset C_r(A)$ is ρ-closed or d-closed if it is a closed subset of a metric space $(C_r(A),\rho)$ or $(\mathcal{K}_r(A),d)$, respectively. In a similar way we define ρ-compact subsets of $C_r(A)$. Finally, using the above mode of d-weak convergence in $\mathcal{K}_r(A)$ we can also define relatively d-weak sequentionally compact subsets of $\mathcal{K}_r(A)$.

1. ρ-Compactness of Attainable Sets

We shall present here some sufficient conditions for the nonemptyness and ρ-compactness of the attainable set $T(D,F,A,B)$ containing all absolutely continuous functions $(x;[\sigma-r,\sigma+a])\in\mathcal{X}_r(A)$ satisfying conditions (1). We begin with the following definitions.

We will say that $F:D\to Conv(\mathbb{R}^n)$ is (u.s.c.w) on D if for every $(t,u,v)\in D$ and every sequence $\{(t_n,u_n,v_n)\}$ of D such that $|t_n-t|\to 0, |u_n-u|_o\to 0$ and $v_n \rightharpoonup v$ as $n\to\infty$, where $|\cdot|_o$ is the supremum norm of $C([-r,0],\mathbb{R}^n)$ and $\rightharpoonup$ denotes the weak convergence in $L([-r,0],\mathbb{R}^n)$ one has $\lim_{n\to\infty} \bar{h}(F(t_n,u_n,v_n),F(t,u,v))=0$ with $\bar{h}(A,B)\triangleq \sup_{a\in A}\inf_{b\in B}|a-b|$ for $A,B\in Conv(\mathbb{R}^n)$.

Given a set $H\subset\mathbb{R}^n$ the contingent cone $T_H(x)$ to H at $x\in H$ we define by

$$T_H(x)\triangleq \bigcap_{\varepsilon>0}\ \bigcap_{\alpha>0}\ \bigcup_{0<h<\alpha}\left[\frac{1}{h}(H-x)+\varepsilon\mathcal{B}\right], \tag{4}$$

where $\mathcal{B}$ is a closed unit ball of $\mathbb{R}^n$.

Finally, given $\Phi\in AC([-r,0],\mathbb{R}^n)$, $l,M>0$ and a nonempty set $H\subset\mathbb{R}^n$ such that $\Phi(0)\in H$ we define a set $K^l_M(\Phi,H)$ by

$$K^l_M(\Phi,H)\triangleq\{\Psi\in AC([-r,0],\mathbb{R}^n):\Psi(0)\in H,\ \int_E|\dot{\Psi}(s)|ds\le\omega^{\Phi}_{lM}(\mu(E))$$
$$\text{for every mesurable set } E\subset[-r,0]\}, \tag{5}$$

where $\omega^{\Phi}_{lM}(\delta)\triangleq\delta(1+M)+\sup_{F\subset[-r,0]}\{\int_F|\dot{\Phi}(s)|ds:\mu(F)\le\delta\}$.

By the above definitions it can be proved the following theorem ([2], Theorem 3).

Theorem 1. Let (D,A) be a conformable pair of nonempty subsets of $R\times C([-r,0],\mathbb{R}^n)\times L([-r,0],\mathbb{R}^n)$ and $R\times AC([-r,0],\mathbb{R}^n)$, respec-

tively and let B be a nonempty closed subset of $R\times C([-r,0],\mathbb{R}^n)\times R\times C([-r,0],\mathbb{R}^n)$. Suppose $F:D\to Conv(\mathbb{R}^n)$ is (u.s.c.w) and bounded on D. Assume A is closed in $R\times C([-r,0],\mathbb{R}^n)$ and relatively weak sequentionally compact in $R\times AC([-r,0],\mathbb{R}^n)$ and has the connected projection, say equal to an interval $[\alpha,\beta]$ on the real line such that $\beta-\alpha>r>0$. Let F,A and B be furthermore such that there are $A_o\subset A,(\sigma,\Phi)\in A_o,a>r,M>0,l>0$ and a compact set $H\subset\mathbb{R}^n$ such that $\Phi(0)\in H,[\sigma,\sigma+a]\times K_M^l(\Phi,H)\subset A_o,A_o\times A_o\subset B,F(D)\not\subset B$ and $F(t,\Psi,\dot{\Psi})\cap T_H(\Psi(0))\neq\emptyset$ for $t\in[\sigma,\sigma+a]$ and $\Psi\in K_m^l(\Phi,H)$ then T(D,F,A,B) is a nonempty relatively d-weakly compact subset of $\mathcal{X}_r(A)$, ρ-compact in $C_r(A)$.

2. Existence of Optimal Trajectory

We shall consider now the problem minimization of the functional $I[x]=g(\sigma,x_\sigma,\sigma+a,x_{\sigma+a})$ on the attainable set T(D,F,A,B). Such problem is called the Mayer problem for systems described by a neutral functional-differential inclusion $\dot{x}(t)\in F(t,x_t,\dot{x}_t)$. It is denoted by $\mathcal{M}(D,F,A,B,g)$.

We shall prove now the following existence theorem for the Mayer problem defined above.

T h e o r e m 2. Let (D,A) be a conformable pair of nonempty subsets of $R\times C([-r,0],\mathbb{R}^n)\times L([-r,0],\mathbb{R}^n)$ and $R\times AC([-r,0],\mathbb{R}^n)$, respectively and let B a nonempty closed subset of $R\times C([-r,0],\mathbb{R}^n)\times R\times C([-r,0],\mathbb{R}^n)$. Assume F,A and B are such as in Theorem 1. Then for every lower semicontinuous function $g:B\to R$ there is an optimal trajectory of the Mayer problem $\mathcal{M}(D,F,A,B,g)$.

P r o o f. Let us observe that A is compact in $R\times C([-r,0],\mathbb{R}^n)$. By Theorem 1, T(D,F,A,B) is a nonempty relatively d-weakly sequentionally compact subset of $\mathcal{X}_r(A)$, ρ-compact in $C_r(A)$. Therefore,

a pair (A,B) has a nonempty intersection property. Then $(A\times A)\cap B$ is a nonempty compact subset of $R\times C([-r,0],\mathbb{R}^n)\times R\times C([-r,0],\mathbb{R}^n)$. Since g is lower semicontinuous on B then by Weierstrass theorem there is a $\gamma>0$ such that $\inf\{g(z):z\in(A\times A)\cap B\}\geq-\gamma$. Hence it follows that for every $x\in T(D,F,A,B)$ we have $I[x]\geq-\gamma$.
Let $i\triangleq\inf\{I[x]:x\in T(D,F,A,B)\}$. We have $i\geq-\gamma>-\infty$. Since $T(D,F,A,B)\neq\emptyset$ then we also have $i<+\infty$. Thus $i\in(-\infty,+\infty)$ and therefore there exists a minimizing sequence, say $\{(x^n;[\sigma_n-r,\sigma_n+a_n])\}$ for i, i.e., there is a sequence (x^n) of $T(D,F,A,B)$ such that $i=\lim_{n\to\infty} I[x^n]$. Since $T(D,F,A,B)$ is relatively d-weakly sequentionally compact in $\mathcal{X}_r(A)$ then there is a subsequence, say $\{(x^k;[\sigma_k-r,\sigma_k+a_k])\}$ of $\{(x^n;[\sigma_n-r,\sigma_n+a_n])\}$ d-weakly convergent to any $(x;[\sigma-r,\sigma+a])\in T(D,F,A,B)$. By the ρ-compactness of $T(D,F,A,B)$ we can well assume that $(\sigma_k,x^k_{\sigma_k},\sigma_k+a_k,x^k_{\sigma_k+a_k})\to(\sigma,x_\sigma,\sigma+a,x_{\sigma+a})$ in $R\times C([-r,0],\mathbb{R}^n)\times R\times C([-r,0],\mathbb{R}^n)$ as $k\to\infty$. By the lower semicontinuity of g it implies $I[x]\triangleq g(\sigma,x_\sigma,\sigma+a,x_{\sigma+a})\leq\lim_{k\to\infty}\inf\, g(\sigma_k,x^k_{\sigma_k},\sigma_k+a_k,x^k_{\sigma_k+a_k})=i$. Since $(x;[\sigma-r,\sigma+a])\in T(D,F,A,B)$, we have also $I[x]\geq i$. Therefore, $I[x]=i$ and the proof is complete.

References

[1] Cesari, L., Optimization Theory and Application. Appl. of Math. 17, Springer-Verlag 1984.

[2] Kisielewicz, M., Subtrajectory integrals of set-valued functions and neutral functional-differential inclusions, II. Funk. Ekv. (in press).

SMOOTH SOLUTIONS OF LINEAR EQUATIONS WITH SCALAR COEFFICIENTS IN A RIGHT INVERTIBLE OPERATOR

D. Przeworska-Rolewicz (Warszawa)

The purpose of the present paper is to examine the smoothness of solutions to linear equations with scalar coefficients in a right invertible operator. This leads to studies of some spaces bearing an analogy to C^∞-spaces and their subspaces, in particular to spaces of analytic functions.

CONTENTS

1. Algebraic properties of D_∞-spaces
2. Spaces of D-analytic elements
3. Spaces of D-paraanalytic elements
4. Systems with scalar coefficients
References.

1. ALGEBRAIC PROPERTIES OF D_∞-SPACES

Let X be a linear space over a field $\mathcal{F}$ of scalars (for instance, $\mathcal{F}=\mathbb{C}$ or $\mathcal{F}=\mathbb{R}$). By L(X) we denote the set of all linear operators with domains and ranges in X. In particular, $L_o(X)=\{A\in L(X):\text{dom } A=X\}$. By R(X) we denote the set of all right invertible operators from L(X)*). For a given $D\in R(X)$ we shall denote by $\mathcal{R}_D$ the set of all its right inverses and by $\mathcal{F}_D$ - the set of all initial operators, i.e.

*) Theory of right invertible operators and its applications can be found in the book [4] of the present author.

$$\mathcal{R}_D = \{R \in L_o(X) : DR = I\}$$

$$\mathcal{F}_D = \{F \in L_o(X) : F^2 = F, FX = \ker D \text{ and } \exists_{R \in \mathcal{R}_D} FR = 0\}$$

We shall assume here and in sequel that ker $D \neq \{0\}$, i.e. the operator D under consideration is right invertible but not invertible. There is a one-to-one correspondance between sets $\mathcal{R}_D$ and $\mathcal{F}_D$. Namely, an operator F is an initial operator for D if and only if there exists an $R \in \mathcal{R}_D$ such that

(1.1) $\quad F = I - RD$ on dom D.

On the other hand, if $F \in L_o(X)$ is <u>any</u> projection <u>onto</u> ker D then F is an initial operator for and the corresponding right inverse $R = R_o - FR_o$ does not depend on the choice of a right inverse R_o.

An operator $A \in L_o(X)$ is said to be a <u>Volterra operator</u> if $I - \lambda A$ is invertible for all $\lambda \in \mathbb{C}$. The set of all Volterra operators acting in $L_o(X)$ will denoted by V(X).

Let $\mathbb{N}$ be the set of all positive integers. Write:

(1.2) $\quad \mathbb{N}_o = \mathbb{N} \cup \{0\}$

(1.3) $\quad D_o = X$, $D_k = \text{dom } D^k$ for $k \in \mathbb{N}$, $D \in R(X)$

(1.4) $\quad D_\infty = \bigcap_{k=0}^{\infty} D_k$.

Clearly,

(1.5) $\quad X = D_o \supset D_1 \supset \ldots \supset D_k \supset \ldots \supset D_\infty$.

If dom $D \neq X$ then $D_k \neq D_{k+1}$ for $k = 0, 1, 2, \ldots + \infty$.

The following spaces were distinguished by von Trotha [7] as <u>canonical</u> for a $D \in R(X)$:

D_∞ itself;

(1.6) $S=\bigcup_{i=0}^{\infty} \ker D^i$;

(1.7) $Q_R=\bigcap_{i=0}^{\infty} R^i X$, $R\in\mathcal{R}_D$.

By definition, S is the set of all D-polynomials:

(1.8) $S=\mathrm{lin}\{R^k z : z\in\ker D,\ k\in\mathbb{N}_0\}$

and this set is independent of the choice of $R\in\mathcal{R}_D$ (cf. [4]).

EXAMPLE 1.1. Let $X=C[a,b]$, $D=\frac{d}{dt}$, $R=\int_{t_0}^{t}$, where $t_0\in[a,b]$. Then $D_k=C^k[a,b]$ for $k\in\mathbb{N}_0$ and $D_\infty=C^\infty[a,b]$.

DEFINITION 1.1 (von Trotha [7]). Let $D\in R(X)$ and let $R\in\mathcal{R}_D$ be arbitrarily fixed. The triple (X,D,R) is called a D-R space. Let Y be a subspace of X. The triple $(Y,D|_Y,R|_Y)$ is a D-R subspace if there is a subspace $Y_1\subset X$ such that $Y_1\subset \mathrm{dom}\, D$, $RY\subset Y$ and $DY_1\subset Y$. If, in addition, $Y_1=Y$, then Y is said to be D-R invariant subspace.

We shall recall without the proof the following

THEOREM 1.1 (von Trotha [7]). Let $D\in R(x)$ and let $R\in\mathcal{R}_D$. Then

(i) S is the smallest D-R invariant subspace of X containing the kernel of D;

(ii) Q_R is the largest D-R invariant subspace of X such that D and R are mutually inverse and

(1.9) $Q_R=\bigcap_{i=0}^{\infty} R^i D_\infty=\{x\in D_\infty : R^n D^n x=x \text{ for all } n\in\mathbb{N}\}$.

Moreover, if $Q_R\neq\{0\}$ then $\dim Q_R=+\infty$.

(iii) $S\subset D_\infty$, $Q_R\subset D_\infty$, $S\cap Q_R=\{0\}$ and $S\oplus Q_R$ is a D-R invariant subspace;

(iv) D_∞ is the largest D-R invariant subspace of X.

THEOREM 1.2. Suppose that $D \in R(X)$ and F is an initial operator for D corresponding to an $R \in \mathcal{R}_D$ (i.e. $FR=0$). Then

(1.10) $\quad Q_R = \tilde{Q}_F$, where $\tilde{Q}_F = \{x \in D_\infty : FD^n x = 0 \text{ for } x \in \mathbb{N}_0\}$.

Proof. Suppose that $x \in Q_R$. Then the Taylor Formula (cf. [4]) and Formula (1.8) together imply that for every $n \in \mathbb{N}$ we have

$$\sum_{k=0}^{n-1} R^k F D^k x = (I - R^n D^n)x = x - R^n D^n x = 0.$$

In particular, $Fx=0$. By an easy induction we find that $R^n F D^n x = 0$ for all $n \in \mathbb{N}$, which implies that $FD^n x = D^n R^n F D^n x = 0$ for all $n \in \mathbb{N}$. Hence $x \in \tilde{Q}_F$. Conversely, suppose that $x \in \tilde{Q}_F$. Then $FD^n x = 0$ for all $n \in \mathbb{N}$, which implies that for all $n \in \mathbb{N}$ $x - R^n D^n x = (I - R^n D^n)x = \sum_{k=0}^{n=1} R^k F D^k x = 0$. Hence $x \in Q_R$.

COROLLARY 1.1. Suppose that D, R, F are as in Theorem 1.1. Then $x \in Q_R$ if and only if $x \in D_\infty$ and $D^n x \in \ker F$ for all $n \in \mathbb{N}$.

COROLLARY 1.2. Suppose that X, D, R, F are as in Theorem 1.1. Write:

(1.11) $\quad Q = \bigcup_{R \in \mathcal{R}_D} Q_R$; $\tilde{Q} = \bigcup_{F \in \mathcal{F}_D} \tilde{Q}_F$.

Then $\tilde{Q} = Q \subset D_\infty$.

The space Q is called space of singular elements.

PROPOSITION 1.1. Suppose that X, D, R, F are as in Theorem 1.1 and that

$$\tilde{Q}_{F,m} = \{x \in D_\infty : \forall_{n \geq m} FD^n x = 0\} \quad \text{for an } m \in \mathbb{N}.$$

If $x \in \tilde{Q}_{F,m}$ then

$$u = x - \sum_{k=0}^{m-1} R^k F D^k x \in Q_R = \tilde{Q}_F,$$

i.e. u is singular.

Proof. By our assumption, since DF=0 and FR=0, we have for $0 \leq j \leq m-1$

$$FD^j u = FD^j x - FD^j \sum_{k=0}^{m-1} R^k FD^k x = FD^j x - \sum_{k=0}^{j} FD^{j-k} FD^k x - \sum_{k=j+1}^{m-1} FR^{k-j} FD^k x$$

$$= FD^j x - F^2 D^j x = FD^j x - FD^j x = 0$$

and for $j \geq m$

$$FD^j u = FD^j x - FD^j \sum_{k=0}^{m-1} R^k FD^k x = FD^j x - \sum_{k=0}^{m-1} FD^{j-k} FD^k x = FD^j x = 0.$$

EXAMPLE 1.2. Let $X=C(\mathbb{R})$, $D=\frac{d}{dt}$, $R=\int_0^t$, $(Fx)(t)=x(0)$ for $x \in X$. Then F is an initial operator for D corresponding to R. Define: $x(t)=e^{-1/t^2}$ for $t \neq 0$ and $x(0)=0$. Then $x \in C^{\infty}[-a,a]$ for an arbitrary $a \in \mathbb{R}_+$ and $x^{(n)}(0)=0$ for all $n \in \mathbb{N}_0$. Hence x is a singular function, more precisely, $x \in Q_R$.

Recall without proof the following

THEOREM 1.3 (cf. the author [4]). Let X be a linear space over an algebraically closed field $\mathcal{F}$ of scalars. Denote by $\mathcal{F}[t]$ the set of all polynomials with scalar coefficients (i.e. with coefficients from $\mathcal{F}$). Let $P(t) \in \mathcal{F}[t]$ and let

$$P(t)=\prod_{j=1}^{n}(t-t_j)^{r_j}, \text{ where } t_j \neq t_k \text{ for } j \neq k, \ r_1+\ldots+r_n=N.$$

Suppose that $D \in R(X)$ and R is a Volterra right inverse of D. Then the general solution of the equation

$$P(D)x=y, \ y \in X \tag{1.12}$$

is of the form

$$x=\sum_{j=1}^{n}(I-t_jR)^{-r_j}\sum_{m=0}^{r_j-1}(d_{jm}R^m y+z_{jm}, \text{ where } z_{jm} \in \ker D) \tag{1.13}$$

and the scalars d_{jm} are determined by decomposition of the rational

function $1/P(t)$ onto vulgar fractions:

$$(1.14)\qquad \frac{1}{P(t)}=\sum_{j=1}^{n}(I-t_jt)^{-r_j}\sum_{m=0}^{r_j-1}d_{jm}t^m.$$

PROPOSITION 1.2. Suppose that $D\in R(X)$ and $R\in\mathcal{R}_D$. If $u\in D_k$ for a $k\in\mathbb{N}_o$ then

(i) $R^mu\in D_{k+m}$ for $m\in\mathbb{N}$ (cf. Formula 1.3);

(ii) $(I-\lambda R)^{-j}u\in D_k$ for $\lambda\in\mathcal{F}$, $j\in\mathbb{N}$, provided that $R\in V(X)$;

(iii) $(I-\lambda R)^{-j}R^mu=R^m(I-\lambda R)^{-j}u\in D_{k+m}$ for $\lambda\in\mathcal{F}$, $j,m\in\mathbb{N}_o$, provided that $R\in V(X)$.

Proof. (i) Since $D(Ru)=DRu=u\in D_k$, we have $Ru\in D_{k+1}$. By an easy induction we find that $R^mu\in D_{k+m}$ for $m\in\mathbb{N}$. (ii) Write: $v=(I-\lambda R)^{-1}u$. Then $u=v-\lambda Rv$. Hence $Du=Dv-\lambda v$ and $Dv=Du+\lambda v$. This implies that $v\in D$. Again, by an easy induction we find that $D^mv=\sum_{j=0}^{m}\lambda^{m-j}D^ju$ for $m\le k\in\mathbb{N}$. Hence $v\in D_k$. By an induction with respect to j we obtain (ii). Points (i) and (ii) together imply (iii).

An imediate consequence of Theorem 1.3 and Proposition 1.1 is

PROPOSITION 1.3. Suppose that all assumptions of Theorem 1.3 are satisfied. If $y\in D_k$ for a $k\in\mathbb{N}_o$ then every solution of Equation (1.12) belongs to D_{k+N-n}, hence to D_k.

Proof. Point (iii) of Proposition 1.1 implies that $x\in D_{k+\alpha}$, where $\alpha=\sum_{j=1}^{n}(r_j-1)=\sum_{j=1}^{n}r_j-n=N-n\ge 0$. By definition, $D_{k+\alpha}\subset D_k$ for every $k,\alpha\in\mathbb{N}_o$.

COROLLARY 1.3. Suppose that all assumptions of Theorem 1.3 are satisfied and that $P(t)$ has single roots only (i.e. $r_1=\ldots=r_n=1$). If $y\in D_k$ for a $k\in\mathbb{N}_o$ then every solution of Equation (1.12) belongs to D_k (and, in general, do not belong to D_{k+1}).

NOTE: Spaces D_k for $k\in\mathbb{N}$ are not D-R invariant.

DEFINITION 1.2 (cf. von Trotha [7]). Let $\mathcal{F}$ be an algebraically closed field of scalars. Let U be a D-invariant subspace of X, i.e. $DU \subset U$. Then U is said to be D-closed if for every $P(t) \in \mathcal{F}[t]$ we have $P(D)x \in U$ if and only if $x \in U$. The set

$$H_D(U) = \{x \in X: \overset{\exists}{P(t) \in \mathcal{F}[t]} P(D)x \in U\} \tag{1.15}$$

is called a D-hull of U.

Clearly, a D-R invariant subspace $U \subset X$ is D-closed if and only if $H_D(U) = U$.

Observe that $H_D(0)$ is the space of all solutions, of all homogeneous equations $P(D)x = 0$ with scalar coefficients.

We recall, again without proof, the following

THEOREM 1.4 (cf. von Trotha [7]). Suppose that $D \in R(X)$ and $R \in V(X) \cap \mathcal{R}_D$. Then for every D-R invariant subspaces $U, V \subset X$ we have:

(i) $U \subset H_D(U) = H_D[H_D(U)]$. Moreover, $H_D(U)$ is a D-R invariant subspace of D_∞;

(ii) $H_D(U+V) = H_D(U) + H_D(V)$ and $H_D(U \cap V) = H_D(U) \cap H_D(V)$;

(iii) Rational functions of R preserve $H_D(U)$;

(iv) $H_D(U)$ is the smallest extension space of U where every equation with scalar coefficients $P(D)x = y \in H_D(U)$ has a solution and all solutions belong to $H_D(U)$;

(v) $H_D(S) = H_D(0)$;

(vi) $H_D(Q_R) = Q_R \oplus H_D(0)$.

NOTE: Spaces D_k for $k \in \mathbb{N}$ are not D-closed.

THEOREM 1.5. Suppose that all assumptions of Theorem 1.3 are satisfied. Then $H_D(D_\infty) = D_\infty$, i.e. D_∞ is D-closed. In other words:

if $y \in D_\infty$ *and* x *satisfies Equation (1.12) then* $x \in D_\infty$.

Proof. By Point (iv) of Theorem 1.1, D_∞ is the largest D-R invariant subspace of X. By Point (iii) of Theorem 1.4 rational functions of R preserve D_∞. Assume that $y \in D_\infty$. Observe that any constant, i.e. an element $z \in \ker D$ belongs to D_∞. This, Proposition 1.2 (iii) and Proposition 1.3 together imply that $x \in D_\infty$. Point (iv) of Theorem 1.4 implies that $H_D(D_\infty) = D_\infty$. Hence D_∞ is D-closed.

We should point out that D_∞ is the largest D-closed subspace of X. If $x \in D_\infty$ then x is said to be smooth.

Recall that x_λ is an exponential for D if $0 \neq x_\lambda \in \ker(D - \lambda I)$ for a $\lambda \in \mathcal{F}$ (cf. the author [4]).

PROPOSITION 1.4. Let X be a linear space over a field $\mathcal{F}$ of scalars and let $D \in R(X)$. Suppose that F is an initial operator for D corresponding to R and $0 \neq \lambda \in \mathcal{F}$. Then

(i) all eigenvectors of R belong to Q_R, provided that they exist;

(ii) $x_\lambda \in Q_R$ is an exponential if and only if $x_\lambda \in \ker(I - \lambda R) \subset Q_R$ (i.e. if x_λ is an eigenvector for R corresponding to the eigenvalue $1/\lambda$, which belongs to Q_R);

(iii) if an exponential $x_\lambda \notin Q_R$ then $R^n F D^n x_\lambda = \lambda^n R^n z$ for all $n \in \mathbb{N}_0$, where $0 \neq z = Fx_\lambda \in \ker D$;

(iv) if $\ker(I - \lambda R) = \{0\}$ then $Fx_\lambda \neq 0$ for any exponential x_λ;

(v) if $\ker(I - \lambda R) \neq \{0\}$ then $Fx_\lambda = 0$ for any exponential x_λ.

Proof. Points (i)-(iii) have been proved by von Trotha [7]. (iv) Suppose that x_λ is an exponential, i.e. $0 \neq x_\lambda \in \ker(D - \lambda I)$. Then $D(I - \lambda R)x_\lambda = (D - \lambda I)x_\lambda = 0$ which implies $(I - \lambda R)x_\lambda = z$, where $z \in \ker D$ and $z = Fz = F(I - \lambda R)x_\lambda = Fx_\lambda - FRx_\lambda = Fx_\lambda$, for $FR = 0$. Suppose that $Fx_\lambda = 0$. Then we have $(I - \lambda R)x_\lambda = z = Fx_\lambda$ which implies that $x_\lambda \in \ker(I - \lambda R) = \{0\}$. Thus $x_\lambda = 0$, which contradicts our assumption. Hence $Fx_\lambda \neq 0$. Point (v) follows from (i) and (ii), because in this case $0 \neq x_\lambda \in \ker(I - \lambda R) \subset Q_R$ implies $FD^n x_\lambda = 0$ for all $n \in \mathbb{N}_0$, in particular $Fx_\lambda = 0$.

This leads us to the following

DEFINITION 1.3. Suppose that $D\in R(X)$, F is an initial operator for D corresponding to an $R\in\mathcal{R}_D$ and x_λ is an exponential. If $1/\lambda$ is a regular value of R then $Fx_\lambda\neq 0$ and x_λ is a regular exponential. If $1/\lambda$ is an eigenvalue of R then $x_\lambda\in Q_R$, hence $Fx_\lambda=0$ and x_λ is a singular exponential.

EXAMPLE 1.3. Let $X=C[0,1]$, $D=\frac{d}{df}$, $(Fx)(t)=\frac{1}{2}[x(0)+x(1)]$ for $x\in X$. Then F is an initial operator for D corresponding to a right inverse $R=\int_0^t-\frac{1}{2}\int_0^1=\frac{1}{2}(\int_0^t+\int_1^t)$. The right inverse R is not a Volterra operator and has λ as an eigenvalue if $Fe^{\lambda t}=0$ (cf. [4]). We find: $0=Fe^{\lambda t}=\frac{1}{2}(1+e^{\lambda})$, i.e. $e^{\lambda}=-1$. If $\mathcal{F}=\mathbb{R}$, there is no $\lambda\in\mathbb{R}$ satisfying this equality. Hence $R\in V(X)$ and functions $e^{\lambda t}$ are regular exponential for all $\lambda\in\mathbb{R}$. If $\mathcal{F}=\mathbb{C}$ then the equation $e^{\lambda}=-1$ has infinitely many roots $\lambda_k=-\pi i+2k\pi i$, where $k\in\mathbb{Z}$. All numbers $1/\lambda_k$ are eigenvalues of R. Therefore all functions $e^{\lambda_k t}$ $(k\in\mathbb{Z})$ are singular exponentials. If $\lambda\neq\lambda_k$ then $e^{\lambda t}$ is a regular exponential.

EXAMPLE 1.4. Let X and D be defined as in Example 1.3. Let $(Fx)(t)=x(0)+x'(t)$. In a similar manner as in the preceeding example we conclude that the corresponding right inverse is $(Rx)(t)=\int_0^t x(s)ds-x(1)$ and that R has a real eigenvalue λ_0, namely, a root of the equation $1+\lambda e^{\lambda}=0$. Hence the function $e^{\lambda_0 t}$ is a singular exponential in the space X considered over the field of reals.

2. SPACES OF D-ANALYTIC ELEMENTS

Here and sequel X will a complete linear metric space with F-norm $\|\ \|$ and $\mathcal{F}=\mathbb{C}$, if it is not stated otherwise.

THEOREM 2.1. Let $D\in R(X)$. Write:

$$A_R(D)=\{x\in D_\infty:\ x=\sum_{n=0}^{\infty}R^nFD^nx \text{ is convergent and } x=\sum_{n=0}^{\infty}R^nFD^nx\} \tag{2.1}$$

where F is an initial operator for D corresponding to an $R \in \mathcal{R}_D$. Then

$$A_R(D)=\{x\in D_\infty : \lim_{n\to\infty} R^n D^n x=0\} \subset D_\infty . \tag{2.2}$$

Proof. Write:

$$Q_n=\sum_{k=0}^{n-1} R^k F D^k, \quad P_n=R^n D^n \text{ for } n\in\mathbb{N} . \tag{2.3}$$

By the Taylor Formula for an arbitrary $x\in D_\infty$ and for $n\in\mathbb{N}$ we have

$$Q_n x=\sum_{k=0}^{n-1} R^k F D^k x=x-R^n D^n x=x-P_n x=(I-P_n)x.$$

Hence

$$x=P_n x+Q_n x \quad \text{for all } n\in\mathbb{N} \text{ and } x\in D_\infty . \tag{2.4}$$

Moreover, both Q_n and P_n are projections. Indeed, $P_n^2=(R^n D^n)(R^n D^n)=R^n(D^n R^n)D^n=R^n D^n=P_n$ and $Q_n^2=(I-P_n)^2=I-P_n=Q_n$ on D_∞ $(n\in\mathbb{N})$.

Suppose now, that $\lim_{n\to\infty} R^n D^n x=0$. Then $\lim_{n\to\infty} Q_n x=\lim_{n\to\infty}(I-P_n)x=\lim_{n\to\infty}(x-R^n D^n x)=x-\lim_{n\to\infty} R^n D^n x=x$, which implies that the series $\sum_{n=0}^{\infty} R^n F D^n x$ is convergent and that $x=\sum_{n=0}^{\infty} R^n F D^n x$. We therefore conclude that $x\in A_R(D)$.

Conversely, if $x\in A_R(D)$ then the series $\sum_{n=0}^{\infty} R^n F D^n x$ is convergent to x. Hence, by the Taylor Formula

$$\lim_{n\to\infty} R^n D^n x=\lim_{n\to\infty}\left(x-\sum_{k=0}^{n-1} R^k F D^k x\right)=x-\lim_{n\to\infty}\sum_{k=0}^{n-1} R^k F D^k x=x-x=0.$$

Write:

$$A(D)=\bigcup_{R\in\mathcal{R}_D} A_R(D) . \tag{2.5}$$

Clearly, by (1.8) we have

$$S \subset A(D) \subset D_\infty . \tag{2.6}$$

The set A(D) is said to be the space of D-analytic elements (cf. the author [5]).

THEOREM 2.2. Let $D \in R(X)$. If $R \in \mathcal{R}_D$ is continuous then $A_R(D)$ is a D-R invariant subspace (cf. [5]).

Proof. Write: $Y = A_R(D)$. We have to prove that $Y \subset \operatorname{dom} D$, $DY \subset Y$, $RY \subset Y$. By definition, $Y \subset D_\infty \subset \operatorname{dom} D$. Let $y \in Y$ be arbitrarily fixed. Then $y \in D_\infty$, $Dy \in D_\infty$, $Ry \in D_\infty$ and $\sum_{k=0}^{\infty} R^k FD^k Dy = \sum_{k=0}^{\infty} R^k FD^{k+1} y = \sum_{j=1}^{\infty} R^{j-1} FD^j y = DR\sum_{j=1}^{\infty} R^j FD^j y = D\sum_{j=1}^{\infty} R^j FD^j y = D(y - Fy) = Dy$ for $DF = 0$. Hence $Dy \in Y$. Similarly,

$$\sum_{k=0}^{\infty} R^k FD^k Ry = FR + \sum_{k=1}^{\infty} R^k FD^{k-1} y = \sum_{j=0}^{\infty} R^{j+1} FD^j = R\sum_{j=0}^{\infty} R^j FD^j y = Ry$$

for $FR = 0$. Hence $Ry \in Y$. The arbitrariness of $y \subset Y$ implies that $DY \subset Y$, $RY \subset Y$.

THEOREM 2.3. Suppose that X is a Banach space, $D \in R(X)$ and there is a quasinilpotent $R \in \mathcal{R}_D$. Then

(i) $H_D[A_R(D)] = A_R(D) \subset A(D)$, hence $A_R(D)$ is D-closed;

(ii) $H_D(0) \subset A_R(D) \subset A(D)$.

In other words: *all solutions of linear equations in D with scalar coefficients and with the right hand side belonging to $A_R(D)$ (hence D-analytic) belong to $A_R(D)$, hence are D-analytic. In particular, all solutions of linear homogeneous equations in D with scalar coefficients are D-analytic* (cf. also [5]).

Proof. (i) By Theorem 1.4 and Formula (2.6), $S \subset A_R(D) \subset H_D[A_R(D)]$ and the D-hull of $A_R(D)$ is preserved by rational functions of R. Consider the equation $P(D)x = y \in A_R(D)$, where P(t) is a polynomial in D with scalar coefficients. Since $y \in A_R(D)$, we have $y = \sum_{n=0}^{\infty} R^n FD^n y$. By our assumptions, R is quasinilpotent, hence a Volterra operator. Thus, by Theorem 1.3, any solution of the equation $P(D)x = y$ can be written in the form (1.13).

Elements of the form $R^j z$, $z \in \ker D$, belong to S, hence to $A_R(D)$. Thus, in order to prove our theorem, we have to show that $Ru \in A_R(D)$, $(I-\lambda R)^{-1}u \in A_R(D)$, whenever $u \in A_R(D)$, $\lambda \in \mathbb{C}$. Having already proved this fact, by an easy induction we obtain that any solution of the equation $P(D)x=y$ belongs to $A_R(D)$, what means that $A_R(D)=H_D[A_R(D)]$. Suppose then that $u \in A_R(D)$, i.e. $u=\sum_{n=0}^{\infty} R^n F D^n u$. By Theorem 2.2, $A_R(D)$ is a D-R invariant subspace. Thus $Ru \in A_R(D)$. Let now $v=(I-R)^{-1}u$. Then, since $FR=0$, we find

$$\sum_{n=0}^{\infty} R^n F D^n v = \sum_{n=0}^{\infty} R^n F D^n (I-\lambda R)^{-1} u =$$

$$= \sum_{n=0}^{\infty} R^n F D^n \sum_{m=0}^{\infty} \lambda^m R^m u = \sum_{m=0}^{\infty} \lambda^m \sum_{n=0}^{\infty} R^n F D^n R^m u =$$

$$= \sum_{m=0}^{\infty} \lambda^m \sum_{n=0}^{\infty} R^n F D^n R^m u =$$

$$= \sum_{n=0}^{\infty} R^n F D^n u + \sum_{m=1}^{\infty} \lambda^m \sum_{n=0}^{\infty} R^n F D^n R^m u =$$

$$= u + \sum_{m=1}^{\infty} \lambda^m [\sum_{n=0}^{m-1} R^n F R^{m-n} u + \sum_{n=m}^{\infty} R^n F D^{n-m} u] =$$

$$= u + \sum_{m=1}^{\infty} \lambda^m R^n F D^{n-m} u = u + \sum_{m=1}^{\infty} \lambda^m \sum_{j=0}^{\infty} R^{m+j} F D^j u =$$

$$= u + \sum_{m=1}^{\infty} \lambda^m R^m \sum_{j=0}^{\infty} R^j F D^j u = u + \sum_{m=1}^{\infty} \lambda^m R^m u = \sum_{m=0}^{\infty} \lambda^m R^m u =$$

$$= (I-\lambda R)^{-1} u = v,$$

what completes the proof of (i).
(ii) By (2.6) we have $S \subset A_R(D)$. This implies that

$$H_D(0) = H_D(S) \subset H_D[A_R(D)] = A_R(D) \subset A(D).$$

PROPOSITION 2.1. Let $D \in R(X)$ and $R \in \mathcal{R}_D$. Then

$$Q_R \cap A_R(D) = \{0\}. \tag{2.7}$$

Proof. Suppose that $x \in Q_R \cap A_R(D)$. Then for every $n \geq 1$ we have $R^n D^n x = x$. On the other hand, since $x \in A_R(D)$, by Theorem 2.1 we

find $0=\lim_{n\to\infty} R^n D^n x=x$. This implies Formula (2.7).

PROPOSITION 2.2. Suppose that $D\in R(X)$. F is an initial operator for D corresponding to a continuous right inverse R. Then R is a Volterra operator on $A_R(D)$.

Proof. Write: $Y=A_R(D)$. Since R is continuous, we have $RY\subset Y$. Suppose that R is not Volterra on Y. Proposition 1.3 implies that there exists a $\lambda\neq 0$ and an exponential $x_\lambda\in\ker(I-\lambda R)\subset Q_R$. This leads to a contradiction, since $x_\lambda\neq 0$ and $Q_R\cap A_R(D)=\{0\}$. Hence $R\in V(Y)$.

Let X be a Banach space. Denote by QN(X) the set of all quasinilpotent operators belonging to $L_o(X)$. Evidently, $QN(X)\subset V(X)$. If the space X is over $\mathbb{C}$ then we have $QN(X)=V(X)$. This implies the following

PROPOSITION 2.3. Let $\dot{X}=C[0,1]$ over $\mathbb{C}$. Then the only quasinilpotent right inverse to the operator $D=d/dt$ are of the form $R=\int_a^t$, where $a\in[0,1]$.

Proof. The only continuous right inverses of D in the space C[0,1] over $\mathbb{C}$ are of the form $R=\int_a^t$, where $a\in[0,1]$ (cf. the author and S. Rolewicz [6]).

In the Banach spaces over reals we have $QN(X)\neq V(X)$. Indeed, if we consider the space C[0,1] over $\mathbb{R}$ and the operator R defined in Example 1.3, then $R\in V(X)$ but $R\notin QN(X)$. However, if we consider the space $Y=X+iX$ and an extension of $A\in L(X)$ defined as follows

$$Ay=Ax_1+i\,Ax_2 \quad \text{for } y=x_1+ix_2, \text{ where } x_1,x_2\in \operatorname{dom} A$$

then we conclude that $QN(X)=V(Y)$.

EXAMPLE 2.1. Suppose that $X=C[1,+\infty)$, $D=t\frac{d}{dt}$, $(Rx)(t)=\int_1^t s^{-1}x(s)ds$, $(Fx)(t)=x(1)$ for $x\in X$. Then $R\in\mathcal{R}_D\cap V(X)$ and

$$x\in A_R(D) \text{ if and only if } x(t)=\sum_{n=0}^{\infty}\frac{(D^n x)(1)}{n!}(\ln t)^n.$$

EXAMPLE 2.2. Let $X=C[0,1]$, $D=\frac{d}{dt}$, $(Fx)(t)=\int_0^1 x(s)ds$. Then the corresponding right inverse is $(Rx)(t)=\int_0^t x(s)ds-\int_0^1(1-s)x(s)ds$. Since $0=Fe^{\lambda t}=\frac{1}{\lambda}(e^{\lambda}-1)$ and $\lambda\neq 0$, we conclude that in the space $C[0,1]$ over reals there is no eigenvalue of R. Hence R is a Volterra operator and all functions $e^{\lambda t}\in A_R(D)$. However, in the space $C[0,1]$ over $\mathbb{C}$ the operator R has eigenvalues $1/\lambda_k$, where $\lambda_k=2\pi i$, $k\in\mathbb{Z}$. Therefore $R\notin V(X)$ and $e^{\lambda_k t}\notin A_R(D)$.

EXAMPLE 2.3. Suppose that $X=(s)$ over $\mathcal{F}$ (where $\mathcal{F}=\mathbb{C}$ or $\mathcal{F}=\mathbb{R}$) with the convergence on each coordinate. Write: $x=\{x_n\}$ where $n\in\mathbb{N}$, $\delta_{m,n}=1$ for $m=n$ and $\delta_{m,n}=0$ otherwise $(m,n\in\mathbb{Z})$, $\delta=\{\delta_{1n}\}$. Consider the operator of forward shift: $D\{x_n\}=\{x_{n+1}\}$. Then the operator D is right invertible, $\ker D$ consists of all sequences $c\cdot\delta$, where $c\in\mathcal{F}$. A right inverse is $R\{x_n\}=\{x_{n-1}\}$ (where we admit $x_{n-k}=0$ if $n\leq k$ for all $x=\{x_n\}\in X$) i.e. the operator of backward shift. By definition, $R^k\{x_n\}=\{x_{n-k}\}$ for $k\in\mathbb{N}$. The corresponding initial operator F is defined as $F\{x_n\}=x_1\delta$. Then $FD^k\{x_n\}=x_{k+1}\delta$, $R^kFD^k\{x_n\}=x_{k+1}R^k\delta=x_{k+1}\{\delta_{n-k,1}\}$ for $k\in\mathbb{N}_0$ and

$$Q_N\{x_n\}=\sum_{k=0}^{N-1}R^kFD^k\{x_n\}=\sum_{k=0}^{N-1}x_{k+1}\{\delta_{n-k,1}\}=$$

$$=\{x_1,x_2,\ldots,x_N,0,0,\ldots\}$$

$$\lim_{n\to\infty}R^ND^N\{x_n\}=\lim_{N\to\infty}R^N\{x_{N+n}\}=\{\underbrace{0,\ldots,0}_{N\text{ coordinates}},x_{N+1},x_{N+2},\ldots\}.$$

By our assumptions, $\lim R^ND^N\{x_n\}=0$ if and only if the n-th coordinate of $R^ND^N\{x_n\}$ $N\to\infty$ tends to zero for all $n\in\mathbb{N}$ as N tends to infinity. Hence

$$\lim_{N\to\infty}R^ND^N\{x_n\}=0 \text{ and } \lim_{N\to\infty}Q_N\{x_n\}=\sum_{k=0}^{N-1}R^kFD^k\{x_n\}=\{x_n\}.$$

The arbitrariness of $x\in X$ implies

$$A_R(D)=X.$$

Proposition 2.2 implies that $R\in V(X)$. It is easy to verify that exponentials are of the form: $c\{\lambda^{n-1}\}$, where $c,\lambda\in\mathcal{F}$.

EXAMPLE 2.4. Suppose that X,D,R are defined as in Example 2.3. Consider an initial operator $F_1\{x_n\}=(x_1+x_2)\delta$. The corresponding right inverse is $R_1\{x_n\}=(R-F,R)\{x_n\}=\{x_{n-1}\}-x_1\delta$. It is easy to verify that all elements of the form $x_\lambda=x_1\{\lambda^{n-1}\}$ belong to $\ker(I-\lambda R_1)$. We have two cases:

1^o $x_1=0$. This implies $x=0$;

2^o $x_1\neq 0$. Then $\lambda=-1$ is an eigenvalue of R_1. Hence $R_1\notin V(X)$ and $A_{R_1}(D)\neq X$. By Proposition 1.3, $\ker(I+R_1)\subset Q_{R_1}$. It can be easily seen that Q_{R_1} consists of all elements of the form $x=x_1\{(-1)^n\}$. Indeed,

$$Q_{R_1}=\{x\in X:\ \overset{\forall}{k\geq 0}\ FD^kx=0\}=\{x\in X:\ \overset{\forall}{k\geq 0}\ (x_{k+1}+x_{k+2})\delta=0\}$$

which implies $x_n=-x_{n-1}=\ldots=(-1)^{n-1}x_1$. Thus

$$Q_{R_1}=\ker(I+R_1).$$

PROPOSITION 2.4. Suppose that $D\in R(X)$, $\ker D\neq\{0\}$, F is an initial operator corresponding to a continuous Volterra right inverse R and $\{S_h\}_{h\in\mathbb{R}_+}$ is the family of D-shifts induced by R, i.e. $S_he_\lambda F=e^{\lambda h}e_\lambda F$, where $e_\lambda=(I-\lambda R)^{-1}$, for all $\lambda\in\mathbb{C}$ for all $\lambda\in\mathbb{C}$, $h\in\mathbb{R}_+$ (cf. the author [3], also [4]). Then all right inverses $R_h=R-FR_h$ are Volterra operators on $A_R(D)$, where $F_h=FS_h$ for $h\in\mathbb{R}_+$.

Proof. By Proposition 2.2, R is Volterra on $A_R(D)$. In order to prove that all R_h are Volterra operators, we have to show that $F_he_\lambda(z)\neq 0$ for $0\neq z\in\ker D$ and $h\in\mathbb{R}_+$, $0\neq\lambda\in\mathbb{C}$ (cf. the author [4] p. 502). Indeed, let $h\in\mathbb{R}_+$ be arbitrarily fixed, $0\neq z\in\ker D, 0\neq\lambda\in\mathbb{C}$. Then $F_he_\lambda(z)=FS_he_\lambda(z)=Fe^{\lambda h}e_\lambda(z)=e^{\lambda h}Fe_\lambda(z)=e^{\lambda h}z\neq 0$.

PROPOSITION 2.5. Suppose that $D \in R(X)$, $\ker D \neq \{0\}$, F is an initial operator corresponding to a continuous right inverse and $\{S_h\}_{h \in \mathbb{R}_+}$ is the family of R-shifts (cf. the author [3], also [4]). Then all right inverses $R_h = R - F_h R$ are Volterra operators on $A_R(D)$, where $F_h = FS_h$ $(h \in \mathbb{R}_+)$.

Proof. Suppose that there exists an $h_o \neq 0$ such that R_{h_o} is not a Volterra operator on $A_R(D)$. This means that there exists a $\lambda_{h_o} \neq 0$ and a $y \in A_R(D)$ such that $0 \neq y \in \ker(I - \lambda_{h_o} R_{h_o}) \subset Q_R$. This implies that $Dy = \lambda_{h_o} y$ and $Fy = 0$ (by Proposition 2.1). Since $R_{h_o} = R - F_{h_o} R = R - FS_{h_o} R$, we find

$$(I - \lambda_{h_o} R)y = \lambda_{h_o} FS_{h_o} Ry = FS_{h_o} RDy = FS_{h_o} (I - F)y = FS_{h_o} y.$$

Hence $0 = F_{h_o} = FS_{h_o} y = (I - \lambda_h R)y$, which contradicts our assumption that R is Volterra operator on $A_R(D)$. Thus all R_h are Volterra operators on $A_R(D)$.

3. SPACES OF D-PARAANALYTIC ELEMENTS

In the present section we shall look for possible extensions of spaces of D-analytic elements which are D-closed.

THEOREM 3.1. Suppose that X is Banach space (over $\mathbb{C}$). $D \in R(X)$ and there is a quasinilpotent $R \in \mathcal{R}_D$. Then

$$H_D[A_R(D) \oplus Q_R] = A_R(D) \oplus Q_R \tag{3.1}$$

i.e. the set $A_R(D) \oplus Q_R$ is D-closed.

In other words: *all solutions of linear equations in D with scalar coefficients and with right hand side which is a sum of a D-analytic element and a singular element, are again sums of a D-analytic element and a singular element.*

Proof. Theorem 2.3 (ii) implies that $H_D(D)\subset A_R(D)$. This, Theorem 2.3 (i), Theorem 1.4 (ii) and (vi), together imply that

$$H_D[A_R(D)+Q_R]=H_D[A_R(D)]+H_D(Q_R)=A_R(D)+Q_R+H_D(D)$$

$$=A_R(D)+Q_R,$$

for both, $A_R(D)$ and Q_R, are D-R invariant (Theorem 2.2 and 2.4). Proposition 2.1 implies that the sum $A_R(D)+Q_R$ is direct.

Write:

(3.2) $$A_R^Q(D)=\{x\in D_\infty : \lim_{n\to\infty} R^nD^nx=x\} \text{ for } R\in\mathcal{R}_D$$

Formulae (2.4) may suggest that any element of D_∞ can be represented as a sum of two elements which belong to $A(D)$ and $A^Q(D)=\bigcup_{R\in\mathcal{R}_D} A_R^Q(D)$, respectively (not necessarily in a unique way). We shall see that the situation in D_∞ is different.

THEOREM 3.2. Suppose that $D\in R(X)$ and F is an initial operator for D corresponding to an $R\in\mathcal{R}_D$. Then

(i) $Q_R\subset A_R^Q(D)\subset D_\infty$;

(ii) if R is continuous then $A_R^Q(D)=Q_R$, i.e. if for an $x\in D_\infty$ we have $\lim_{n\to\infty} R^nD^nx=x$ then $R^nD^nx=x$ for all $n\in\mathbb{N}$.

Proof. (i) By definition, if $x\in Q_R$ then $R^nD^nx=x$ for all $n\in\mathbb{N}$. Hence $\lim_{n\to\infty} R^nD^nx=x$ and $x\in A_R^Q(D)$. (ii) Let $R\in\mathcal{R}_D$ be continuous. Write: $Y=A_R^Q(D)$. Thus Y is a D-R invariant subspace. Indeed, if $y\in Y$ is arbitrarily fixed then

$$\lim_{n\to\infty} R^nD^nDy=\lim_{n\to\infty} R^nD^{n+1}y=DR\lim_{n\to\infty} R^nD^{n+1}y=D\lim_{n\to\infty} R^{n+1}D^{n+1}y=Dy;$$

$$\lim_{n\to\infty} R^nD^ny=\lim_{n\to\infty} R^nD^ny=R\lim_{n\to\infty} R^{n-1}D^{n-1}y=Ry.$$

Thus $Dy, Ry \in Y$. The arbitrariness of $y \in Y$ implies $DY \subset Y$, $RY \subset Y$. By definition, for arbitrary $x \in Y$ we have

$$\lim_{n\to\infty} \sum_{k=0}^{n} R^k F D^k x = 0. \tag{3.3}$$

Indeed, by Taylor Formula,

$$\lim_{n\to\infty} \sum_{k=0}^{n} R^k F D^k x = \lim_{n\to\infty} (x - R^{n+1} D^{n+1} x) = x - \lim_{n\to\infty} R^{n+1} D^{n+1} x = 0.$$

Suppose now that $x \neq 0$ and $x \in Y \setminus Q_R$. This means that there exists an index n such that $FD^n x \neq 0$. Denote by n_o the minimal index with this property. This, and Formula (3.4) together imply that

$$0 = \lim_{n\to\infty} \sum_{k=0}^{n} R^k F D^k x = R^{n_o} F D^{n_o} x + \lim_{n\to\infty} \sum_{k=n_o+1}^{\infty} R^k F D^k x$$
$$= R^{n_o} F D^{n_o} x.$$

Hence $FD^{n_o} x = D^{n_o}(R^{n_o} F D^{n_o} x) = 0$, which contradicts our assumption that n_o is a minimal index such that $FD^{n_o} x = 0$. We therefore conclude that $x = 0$ and $A_R^Q(D) = Q_R$.

Observe that already distinguished D-closed subspaces of D_∞:

$$A_R(D), \quad Q_R, \quad A_R(D) \oplus Q_R$$

have the common property:

$$\lim_{n\to\infty} R^n F D^n x = 0 \tag{3.5}$$

for x belonging to these spaces. Indeed, if $x \in A_R(D)$ then the series $\sum_{k=0}^{\infty} R^n F D^n x$ is convergent. Hence its n-th term $R^n F D^n x$ tends to zero as $n \to \infty$. If $x \in Q_R$ then $FD^n x = 0$ for $n \in \mathbb{N}_o$. Hence $R^n F D^n x = 0$ for $n \in \mathbb{N}_o$. If $x \in A_R(D) \oplus Q_R$ then $x = u + v$, where $u \in A_R(D)$, $v \in Q_R$ and

$$\lim_{n\to\infty} R^n F D^n x = \lim_{n\to\infty} R^n F D^n u + \lim_{n\to\infty} R^n F D^n v = 0.$$

Another example with this property is following

EXAMPLE 3.1. Let $D\in R(X)$ and $R\in\mathcal{R}_D$ be given. Let $Y=X_1\times X_2$, where $X_1=A_R(D)$, $X_2=Q_R$. For an $A\in L(X)$ and $x_1,x_2\in\operatorname{dom} A$ we admit: $A(x_1,x_2)=(Ax_1,Ax_2)$. Let $0\neq x_1\in A_R(D)$, $0\neq x_2\in Q_R$. Then $(x_1,x_2)\neq 0$ and

$$\lim_{n\to\infty} R^nFD^n(x_1,x_2)=(\lim_{n\to\infty} R^nFD^nx_1,\ \lim_{n\to\infty} R^nFD^nx_2)=0.$$

Observe that in this case we have

$$\lim_{n\to\infty} R^nD^n(x_1,x_2)=(\lim_{n\to\infty} R^nD^nx_1,\ \lim_{n\to\infty} R^nD^nx_2)=(0,x_2)\neq(x_1,x_2).$$

Write:

$$PA_R(D)=\{x\in D_\infty:\lim_{n\to\infty} R^nFD^nx=0\} \tag{3.5}$$

where F is an initial operator for $D\in R(X)$ corresponding to an $R\in\mathcal{R}_D$, and

$$PA(D)=\bigcup_{R\in\mathcal{R}_D} PA_R(D). \tag{3.6}$$

Any element of $PA(D)$ is said to be *D-paraanalytic*. Clearly, we have

$$A_R(D),\ Q_R,\ A_R(D)\oplus Q_R\subset PA_R(D)\subset PA(D)\subset D_\infty. \tag{3.7}$$

Thus any D-analytic element is D-paraanalytic, but not conversely. Also any singular element and a sum of singular and D-analytic elements are D-paraanalytic.

THEOREM 3.3. Suppose that $D\in R(X)$ and F is an initial operator for D corresponding to a continuous right inverse R. Then $PA_R(D)$ is a D-R invariant subspace.

Proof. Write: $Y=PA_R(D)$ and let $y\in Y$ be arbitrarily fixed. Then

$$\lim_{n\to\infty} R^nFD^n(Dy)=DR\lim_{n\to\infty} R^nFD^{n+1}y=D\lim_{n\to\infty} R^{n+1}FD^{n+1}y=0;$$

$$\lim_{n\to\infty} R^n FD^n(Ry) = \lim_{n\to\infty} R^n FD^{n-1} y = R\lim_{n\to\infty} R^{n-1} FD^{n-1} y = 0.$$

Thus $Dy\in Y$, $Ry\in Y$. The arbitrariness of $y\in Y$ implies $DY \subset Y$, $RY \subset Y$. Hence Y is D-R invariant.

THEOREM 3.4. Suppose that X is a Banach space, $D\in R(X)$ and F is an initial operator for D corresponding to a quasinilpotent $R\in \mathcal{R}_D$. Then

$$H_D[PA_R(D)] = PA_R(D) \tag{3.8}$$

i.e. the set $PA_R(D)$ is D-closed.

Proof. Formula (3.7) and Theorem 2.3 together imply that $H_D(0) \subset A_R(D) = H_D(A_R(D)) \subset H_D(PA_R(D))$.

In order to prove our theorem it is enough to show that $y\in Y = PA_R(D)$ implies $Ry\in Y$, $(I-\lambda R)^{-1} y\in Y$.

Indeed, by Theorem 3.3, the space Y is D-R invariant. Hence $Ry\in Y$. By an easy induction we find $R^k y\in Y$ for all $k\in\mathbb{N}$. Since, by our assumption, R is quasinilpotent, we have

$$(I-\lambda R)^{-1} = \sum_{k=0}^{\infty} \lambda^k R^k \quad \text{for all} \quad \lambda\in\mathbb{C}.$$

Hence $u=(I-\lambda R)^{-1} y\in Y$.

EXAMPLE 3.2. Let $D\in R(X)$ and let F_1 and $F_2\neq F_1$ be initial operators for D corresponding to right inverses R_1 and R_2, respectively. Let $Y=X_1 \oplus X_2$, where $X_i = A_{R_i}(D) \oplus Q_{R_i}$ $(i=1,2)$. Define:

$$D(x_1,x_2) = (Dx_1, Dx_2) \quad \text{for } x_1,x_2\in \operatorname{dom} D;$$

$$R(x_1,x_2) = (R_1x_1, R_2x_2);\ F(x_1,x_2) = (F_1x_1, F_2x_2) \quad \text{for } x_1,x_2\in X.$$

It is easy to verify that $D \in R(Y)$ and F is an initial operator for D corresponding to $R \in \mathcal{R}_D$. Moreover, if $x_i = u_i + v_i$, $u_i \in A_{R_i}(D)$, $v_i \in Q_{R_i}$ for $i=1,2$, then

$$\lim_{n\to\infty} R^n F D^n(x_1,x_2) = (\lim_{n\to\infty} R_1^n F_1 D^n(u_1+v_1), \lim_{n\to\infty} R_2^n F_2 D^n(u_2+v_2)) = 0.$$

Hence $(x_1,x_2) \in PA_R(D)$. We also have

$$\lim_{n\to\infty} R^n D^n(x_1,x_2) = (\lim_{n\to\infty} R_1^n D^n(u_1+v_1), \lim_{n\to\infty} R_2^n D^n(u_2+v_2)) =$$

$$= (v_1,v_2)$$

and, in general, $0 \neq (v_1,v_2) \neq (x_1,x_2)$.

Theorem 3.4 and Theorem 1.4 (ii) immediately imply the following

COROLLARY 3.1. Let X be a Banach space and let $D \in R(X)$. Write:

$$PA^0(D) = \sum_{R \in QN(X)} PA_R(D), \tag{3.9}$$

i.e. $PA^0(D)$ is an algebraic sum of $PA_R(D)$ extended over all quasinilpotent right inverses R. Then

$$H_D[PA^0(D)] = PA^0(D) \subset PA(D) \tag{3.10}$$

i.e. the set $PA^0(D)$ is D-closed.

In other words: *any solution of a linear equation in D with scalar coefficients and with right-hand side belonging to $PA^0(D)$, hence D-paraanalytic, belongs again to $PA^0(D)$, hence is also D-paraanalytic.*

Proposition 2.4 (Proposition 2.5, respectively) implies

COROLLARY 3.2. Suppose that X is a Banach space $D \in R(X)$, F is an initial operator for D corresponding to a quasinilpotent right

inverse R, $\{S_h\}_{h\in\mathbb{R}_+}$ is either the family of D-shifts or the family of R-shifts. Write $R_h=R-F_hR$, $F_h=FS_h$ for $h\in\mathbb{R}_+$ and

$$PA\hat{}(D)=\sum_{h\in\mathbb{R}_+} PA_{R_h}(D)$$

i.e. $PA\hat{}(D)$ is an algebraic sum of $PA_{R_h}(D)$ for $h\in\mathbb{R}_+$. Then

$$H_D[PA\hat{}(D)]=PA\hat{}(D)\subset PA^0(D)$$

i.e. the set $PA\hat{}(D)$ is D-closed.

Indeed, all R_h are Volterra operators, hence quasinilpotent (for the field of scalars is $\mathbb{C}$). Recall that in Banach spaces R-shifts and D-shifts induced by a quasinilpotent right inverse R do coincide (cf. the author [3], also [4]).

EXAMPLE 3.3. Let $X=C[0,1]$ over $\mathbb{C}$, $D=\frac{d}{dt}$, $R=\int_0^t$. Then $R_h=\int_0^t-\int_0^h=\int_0^t$, $(F_hx)(t)=x(h)$ for $x\in X$, $h\in\mathbb{R}_+$. It means that $PA\hat{}(D)$ consists of all functions such that $\lim_{n\to\infty}\frac{x^{(n)}(h)}{n!}(t-h)^n=0$ for an $h\in\mathbb{R}_+$. In particular, $PA\hat{}(D)$ consists of functions which are either analytic or singular at points $h\in\mathbb{R}_+$ and their sums.

Observe that the set $PA\hat{}(D)$ contains quasianalytic functions in the Mandelbroit sense (cf. [2]).

NOTE: We have assumed in several statements that a right inverse R under consideration is continuous. It is well-known that if $D\in R(X)$, ker D is closed and an $R\in\mathcal{R}_D$ is continuous, then D is closed. The converse statement is not true, as it is shown by the following

EXAMPLE 3.4. Let $X=C[0,1]$ with norm sup and let $D=\frac{d}{dt}$. Hence D is closed and ker D is closed. However, a right inverse $R=\int_0^t+\varphi$, where φ is an *unbounded* linear functional, is not bounded (hence not continuous).

EXAMPLE 3.5. The function

$$x(t)=\sum_{n=1}^{\infty} e^{-(n!)^{1/2}}\cos(n!t)$$

is infinitely differentiable on the interval $[0,2\sigma]$ and not analytic on $[0,2\pi]$ (cf. Mandelbrojt [2]).
We have for $p\in\mathbb{N}_0$

$$x^{(2p)}(t)=(-1)^p\sum_{h=1}^{\infty}e^{-(n!)^{1/2}}(n!)^{2p}\cos(n!t);$$

$$x^{(2p+1)}(t)=(-1)^p\sum_{h=1}^{\infty}e^{-(n!)^{1/2}}(n!)^{2p+1}\sin(n!t).$$

Let $R=\int_0^t$, then $(Fx)(t)=x(0)$. We have $x^{(2p+1)}(0)=0$ for all $p\in\mathbb{N}_0$. However $x^{(2p)}(0)=(-1)^p\sum_{n=1}^{\infty}e^{-(n!)^{1/2}}(n!)^{2p}\neq 0$. Hence $x\notin Q_R$.

J. Rogulski proved that

$$\left|\frac{x^{(2p)}(0)}{(2p)!}\,t^{2p}\right|\to\infty \text{ as } p\to\infty \text{ for arbitrary small } t\neq 0$$

(private communication, February, 1987).

Since in our case $R^{2p}FD^{2p}x=\frac{x^{(2p)}(0)}{(2p)!}\,t^{2p}$, we conclude that the function $x\in C^{\infty}[0,2\pi]$ is not $\frac{d}{dt}$-paraanalytic. The proof is too complicated to be given here.

EXAMPLE 3.6. J. Rogulski proved the following theorem: Let Y be a Banach space over $\mathbb{R}$ and let the set $I\subset\mathbb{R}$ be open and connected. Suppose that $f:I\to Y$ is a function of the class C^{∞} which is analytic on $I\setminus\{a\}$, where $a\in I$. Suppose that for a $t_0\neq a$ we have

$$\lim_{n\to\infty}\frac{1}{n!}(t_0-a)^n f^{(n)}(a)=0.$$

Let $I_0=I\cap(a-\delta,a+\delta)$, where $\delta=|t_0-a|$. Then there exist functions $f_1:I_0\to Y$ and $f_2:I_0\to Y$ such that

(i) f_1 is analytic, f_2 is of the class C^{∞} and analytic in $I_0\setminus\{a\}$. Moreover, $f_2^{(n)}(a)=0$ for all $n\in\mathbb{N}_0$;

(ii) $f(t)=f_1(t)+f_2(t)$ for all $t\in I_o$
(private communication, February, 1987).

Let $X=C(I\to Y)$, $D=\frac{d}{dt}$, $R=\int_a^t$, $(Fx)(t)=x(a)$ for $x\in X$. Then $D\in R(X)$, $R\in QN(X)$ and the Rogulski's theorem says that *every* $\frac{d}{dt}$*-paraanalytic function in* $C(I\to Y)$ *induced by* R *is a sum of an analytic function and a singular function.*

On the other hand, as Rogulski has shown, there exist $\frac{d}{dt}$-paraanalytic functions which cannot be decomposed in this way. Namely, there exists a function $f:I\to\mathbb{R}$, where $I=(-1,1)$ with the properties:

(i) f is of the class C^∞ and bounded;
(ii) f is analytic on $I\setminus\{0\}=(-1,0)\cup(0,1)$;
(iii) $f(0)=1$, $f^{(n)}(0)=2^n n!$ for all $n\in\mathbb{N}$;
(iv) f cannot be decomposed on I into a sum $f=f_1+f_2$, where f_1 is analytic on I and f_2 is of the class C^∞ on I with the property that there exists an $m\in\mathbb{N}$ such that $f_2^{(n)}(0)=0$ for all $n\geq m$.

By Proposition 1.1 a C^∞-function f_2 with the property that there is an $m\in\mathbb{N}$ such that $f_2^{(n)}(0)=0$ for all $n\geq m$ is a difference of a singular function and a polynomial of degree $m-1$ (i.e. an analytic function).

Having already constructed the function f we define: $g(t)=f(\frac{2}{\pi}\arctan t)$. We conclude that $g\in C^\infty(\mathbb{R})$, g is analytic outside of zero and bounded. Moreover, g cannot be written as a sum: $g=g_1+g_2$, where g_1 is analytic on $\mathbb{R}$ and g_2 is of the class C^∞ and has the property that there exists $m\in\mathbb{N}$ such that $g_2^{(n)}(0)=0$ for $n\geq m$. This example was also given by Rogulski. The proofs of these facts are too complicated to be stated here.

4. SYSTEMS WITH SCALARS COEFFICIENTS

The results of the previous sections can be extended for systems

of equations in D with scalar coefficients:

(4.1) $D\xi_j=\sum_{k=1}^{m} a_{jk}\xi_k+\eta_k, \xi_k\in X$ $(k=1,\ldots,m)$,

where $D\in R(X)$ and $a_{jk}\in\mathcal{F}$ for $j,k=1,\ldots,m$.

Indeed, the system (4.1) can written in the following form:

(4.2) $(D-A)x=u, \quad y\in Y,$

where Y is the Cartesian product of m copies of the space X, $x=(\xi_1,\ldots,\xi_m)$, $y=(\eta_1,\ldots,\eta_m)$, $\xi_1,\ldots,\xi_m,\eta_1,\ldots,\eta_m\in X$ and for an arbitrary $T\in L(X)$ we define

(4.3) $Tx=(T\xi_1,\ldots,T\xi_m)$ for $\xi_1,\ldots,\xi_m\in \mathrm{dom}T$.

Then $D\in R(Y)$. Let A be a linear operator induced by the matrix

(4.4) $A=(a_{jk})$ $j,k=1,\ldots,m.$

Clearly, $A\in L_o(Y)$ and A is <u>stationary</u>, i.e. $DA=AD$ on $\mathrm{dom}\, D\subset Y$ and $RA=AR$ on Y, where $R\in\mathcal{R}_D$. The operator A is also <u>algebraic</u>. Denote by P(t) the characteristic polynomial of A:

(4.5) $P(t)=\sum_{k=0}^{N} p_k t^k=\prod_{j=1}^{n}(t-t_j)^{r_j}$, where $p_N=1$, $r_1+\ldots+r_n=N$, $t_j\neq t_k$ for $j\neq k$.

By definition, $P(A)=0$ on Y. Thus Y is a direct sum

(4.6) $Y=Y_1\oplus\ldots\oplus Y_n$, where $Y_j=\ker(A-t_jI)^{r_j}$ $(j=1,\ldots,n)$.

The decomposition (4.6) determines n disjoint projectors P_j giving a <u>partition of unity</u>:

(4.7) $\sum_{j=1}^{n} P_j=I$, $P_jP_k=\delta_{jk}P_j$ $(j,k=1,\ldots,n)$

Since the operator A is stationary, all spaces $Y_1,\ldots,Y_n$ are invariant under A:

(4.8) $\quad AY \subset Y_j \quad (j=1,\ldots,n)$

and the projectors $P_1,\ldots,P_n$ are stationary.

We therefore conclude that Equation (4.2) is equivalent to n independent equations, each one with a polynomial in D with scalar coefficients:

(4.9) $\quad P(D)x_j=\tilde{y}_j$, where $x_j=P_jx$, $y_j=P_jy\in \mathrm{dom}D^{N-1}$

(4.10) $\quad \tilde{y}_j=p_oy_j+\sum_{k=1}^{N}p_kA_{k-1}(D)y_j \quad (j=1,\ldots,n)$

(4.11) $\quad A_k(D)=\sum_{j=0}^{k} A^{k-j}D^j \quad (k\in\mathbb{N}_o)$.

Indeed, by the decomposition (4.6) we find that Equation (4.2) is equivalent n indepent equations:

(4.12) $\quad (D-A)x_j=y_j$, where $x_j=P_jx$, $y_j=P_jy$ $(j=1,\ldots,n)$,

each one considered on the space Y_j satisfying Condition (4.8). By an easy induction we find

$$D^kx_j=A^kx_j+A_{k-1}(D)y_j,\ y_j\in\mathrm{dom}D^{k-1} \quad (j=1,\ldots,n,\ k\in\mathbb{N}).$$

Thus we have, for $P(A)=0$ on Y,

$$P(D)x_j=P(A)x_j+p_oy_j+\sum_{k=1}^{N}p_kA_{k-1}(D)y_j=\tilde{y}_j \quad (j=1,\ldots,n)$$

(cf. the author [4]).

THEOREM 4.1. Suppose that $D\in R(X)$, F is an initial operator for D corresponding to an R, A is stationary (i.e. DA=AD on dom D, RA=AR) and algebraic with the characteristic polynomial P(t).

Suppose that $\tilde{y}_j$ ($j=1,\ldots,n$) determined by Formulae (4.10)-(4.11) belong to D_∞ (respectively: to Q_R). Then all solutions of Equation (4.2) belong to D_∞ (respectively: to $Q_R \oplus H_D(0)$).

Proof. First we have to show that $AD_\infty \subset D_\infty$ and $AQ_R \subset Q_R$. Indeed, if $u \in D_\infty$ then there exists $D^n u$ for all $n \in \mathbb{N}$. This implies that $D^n Au = AD^n u$ for all $n \in \mathbb{N}$. Hence $Au \in D_\infty$. If $u \in Q_R$ then $R^n D^n u = u$ for all $n \in \mathbb{N}$. Since A is stationary, we find $Au = AR^n D^n u = R^n D^n Au$ for all $n \in \mathbb{N}$. Thus $Au \in Q_R$. If $\tilde{y}_j \in D_\infty$ (respectively: $\tilde{y}_j \in Q_R$) for $j=1,\ldots,n$ then the j-th equation (4.2) has all solutions belonging to D_∞ (respectively: to $Q_R \oplus H_D(0)$; cf. Theorem 1.5 and 1.4 (vi), respectively). Since x is a solution of Equation (4.2) if and only if $x = x_1 + \ldots + x_n$, where x_j is a solution of the j-th equation (4.9), we conclude that x belongs to D_∞ (respectively: to $Q_R \oplus H_D(0)$).

THEOREM 4.2. Suppose that X is a complete linear metric space (over $\mathbb{C}$), $D \in R(X)$, F is an initial operator for D corresponding to a continuous $R \in \mathcal{R}_D$, A is stationary, continuous and algebraic with the characteristic polynomial P(t). Then

(i) the spaces $A_R(D)$, $A_R(D) \oplus Q_R$, $PA_R(D)$ are invariant under A;
(ii) if X is a Banach space, R is quasinilpotent and $\tilde{y}_j$ ($j=1,\ldots,n$) determined by Formulae (4.10)-(4.11) belong to $A_R(D)$ (respectively: to $A_R(D) \oplus Q_R$ to $PA_R(D)$) then all solutions to Equation (4.2) belong to $A_R(D)$ (respectively: to $A_R(D) \oplus Q_R$, to $PA_R(D)$).

Proof. (i) Suppose that $u \in A_R(D)$. Then $\lim_{n\to\infty} R^n D^n u = 0$. Since A is stationary and continuous, we find $\lim_{n\to\infty} R^n D^n Au = \lim_{n\to\infty} AR^n D^n u = A \lim_{n\to\infty} R^n D^n u = 0$. Hence $Au \in A_R(D)$. This, and the corresponding part of the proof of Theorem 4.1, together imply that the set $A_R(D) \oplus Q_R$ is invariant under A. Suppose that $u \in PA_R(D)$. Then $\lim_{n\to\infty} R^n FD^n u = 0$. Hence $\lim_{n\to\infty} R^n FD^n Au = \lim_{n\to\infty} A\, R^n FD^n u = A \lim_{n\to\infty} R^n FD^n u = 0$, what means that $Au \in PA_R(D)$. The proof of Point (ii) is going on the same lines as the proof of Theorem 4.1, if we apply Theorems 2.3, 3.1, 3.4, respectively.

Observe that without any essential change of the proof of Theorem 4.2 (ii) we obtain

THEOREM 4.3. Suppose that X is a Banach space (over $\mathbb{C}$), $D \in R(X)$, F_j are initial operators for D corresponding to quasinilpotent right inverses $R_j \in \mathcal{R}_D$, A is stationary, continuous and algebraic with the characteristic polynomial P(t). Suppose that for $j=1,\ldots,n$

$$E_j = D_\infty \text{ or } A_{R_j}(D) \text{ or } A_{R_j}(D) \oplus Q_R \text{ or } PA_{R_j}(D).$$

If $\tilde{y}_j$ $(j=1,\ldots,n)$ determined by Formulae (4.10)-(4.11) belong to E_j then all solutions of Equation (4.2) belong to the set $E=E_1+\ldots+E_n$.

In a similar manner we can obtain solutions of Equation (4.2) belonging either to $PA^{\circ}(D)$ or $PA\hat{}(D)$ (cf. Corollary 3.1 and 3.2, respectively).

REFERENCES

[1] H. Lausch and D. Przeworska-Rolewicz, Pseudocategories, Paraalgebras and Linear Operators (to appear in Math. Nachrichten).

[2] S. Mandelbrojt, Analytic Functions and Classes of Infinitely Differentiable Functions, The Rice Insitute Pamphlet, XXIX, 2, Houston, Texas, 1942.

[3] D. Przeworska-Rolewicz, Shifts and Peridodicity for Right Invertible Operators, Research Notes in Mathematics, 47, Pitman, London, 1980.

[4] -"-, Algebraic Analysis, PWN-Polish Scientific Publishers and D. Reidel, Warszawa-Dordrecht, 1987.

[5] -"-, Remarks on D_∞-spaces, Proc.Intern.Conf. "Functions Spaces", held in Poznan, August 1986 (to appear).

[6] -"-, and S. Rolewicz, On a Representation of Continuous Volterra Right Inverses to the Derivative in the Spaces of Infinitely Differentiable Functions, J. Austral. Math. Soc. 41 (1986), 138-142.

[7] H. von Trotha, Structure Properties of D-R spaces, Dissertationes Math. CLXXXIV, Warszawa, 1981.

Institute of Mathematics
Polish Academy of Sciences
Sniadeckich 8
00-950 Warszawa
Poland

V. Applications

PRODUCTION-POLLUTION CYCLES

Gustav Feichtinger

Institute for Econometrics and Operations Research
University of Technology
Argentinierstraße 8, A-1040 Vienna, Austria

ABSTRACT. The paper introduces the concept of a firm's "normal" pollution level as a weighted average of past "actual" pollution levels. The impact of an incentive scheme is analyzed, in which the representative firm pays a penalty (or receives a subsidy) when it deviates above (or below) its normal pollution level. The conclusions is, a cyclical production and pollution pattern may be optimal under certain conditions.

1. INTRODUCTION

There are two broad classes of instruments for pollution control. The first class is governmental regulation. It includes prohibition of certain uses, prescribing protective measures (e.g. installing smog-suppressors on cars) or setting limits to the amounts of effluent factories can discharge into a stream (environmental standards). The second class is economic or financial inducement: imposing taxes or changes, or offering subsidies to discourage or encourage different ways of using the resource, leaving the users free to decide how best to respond to the inducement.

The present paper deals with the second class of instruments. We consider a representative firm which has a certain amount of emission "rights". The firm can use these emission rights itself, or sell them to other firms (emission trading), i.e. there is a market of emission certificates[1].

The government or the pollution control authority fixes both the selling and purchase price of certificates. The main purpose of the following study is to analyze the impact of an *incentive scheme* in which a firm receives a reward (or pays a penalty) when it deviates below (above) a "normal" pollution level. The result is that such an institutional setting may imply a cyclical production-pollution trade-off.

---

1) As example we can quote the so-called "bubble-concept" in the air-pollution control in the USA (Siebert, 1982). For a general discussion of environmental control we refer to Wicke (1982), Baumol and Oates (1975).

The paper is organized as follows. Extending a model of Forster (1977) in Section 2 a two state variable pollution control model is formulated. In Section 3 the necessary optimality conditions of a maximum principle are stated, and the resulting canonical system is analyzed numerically. In Section 4 the solutions are presented and interpreted economically. Finally, in Section 5 some conclusions are drawn and possible extensions are discussed.

2. A TWO STATE VARIABLE POLLUTION CONTROL MODEL

Forster (1977) considers a one state variable pollution control problem. Focusing solely on the consumption-pollution trade-off, he assumes that a fixed output is produced in each time period. This output is to be allocated to consumption or to pollution control (abatement). The society derives utility from consumption but in consuming output, a stock of pollution is accumulated which generates disutility. A central planner intends to maximize the discounted utility/disutility stream over an infinite planning horizon. Using standard optimal control theory, Forster (1977) presents conditions under which it is either optimal to eliminate pollution or it is not. For both cases he carries out a phase portrait analysis in the state-costate phase plane, which yields interesting insight into the transient solution, depending on the initial pollution level.

Feichtinger and Luptacik (1987) investigate a pollution control problem for a profit maximizing firm, from the point of view of employment and environmental policy. They define an "actual" emission rate, E, which is influenced both by the production and the abatement activities. Moreover, they consider also a "normal" (standard) emission level, $\bar{E}$, assumed to be constant, which is compared with E. If E exceeds $\bar{E}$, penalty costs are charged to the firm, while for E smaller than $\bar{E}$, the representattive firm obtains a financial reward.

The present paper combines both approaches sketched above in a microeconomic framework. The following extensions are made: First, the emission rate E is replaced by a stock of pollution P. Second, rather than to consider a constant exogenously given, "normal" pollution level $\bar{P}$, the environmental standard *(normal pollution level)* is defined as an exponentially weighted average of past pollution levels as follows:

$$R(t) = m\int_{-\infty}^{t} e^{-m(t-s)}P(s)ds. \qquad (1)$$

The parameter $m > 0$ represents the relative weight given to more recent pollution levels. By differentiating (1) with respect to time, t yields[1)]

---

1) A similar adjustment dynamics is considered in Schmalensee (1972), Ryder and Heal (1973), Long and Siebert (1985), Steindl et al. (1986), and Luhmer et al. (1987).

$$\dot{R} = m(P-R), \tag{2}$$

where here and in the following, the time arguments are omitted. For a given initial normal level of pollution,

$$R(0) = R_0, \tag{3}$$

and an actual pollution path $P(\cdot)$ the differential equation (2) defines an adjustment mechanism in the sense that

$$\dot{R} \gtreqless 0 \quad \text{for} \quad P \gtreqless R.$$

Moreover, both pollution levels have identical stationary values.

The *actual level of pollution*, P[1], is accumulated according to

$$\dot{P} = Z(u) - \delta P, \; P(0) = P_0. \tag{4}$$

Here δ is the natural cleaning rate, and u denotes the rate of production of the representative firm. There is a maximal production rate $\bar{u}$ such that

$$0 \leqq u \leqq \bar{u}. \tag{5}$$

Z(u) determines the contribution of pollution generated by the production rate u. It seems reasonable to assume that Z(u) is convex with

$$Z'(u) > 0, \; Z''(u) > 0. \tag{6a}$$

Moreover, there exists a pollution-neutral level of production, $u_0 \varepsilon (0,\bar{u})$ such that

$$Z(u) \lesseqgtr 0 \quad \text{for} \quad u \lesseqgtr u_0. \tag{6b}$$

Here it is assumed that low levels of production make it possible to carry out abatement activities, by using partly idle production capacities. Similar to Forster (1977), the firm's utility is influenced in a *two-fold* manner by the production rate. First, a higher production level clearly generates more pollution which yields disutility now and in the future. Moreover, each amount of capacity used for production can not be used to clean up pollution.

The level of pollution which is accumulated by Z(u) is assumed to be non-negative:

$$P \geqq 0. \tag{7}$$

The utility rate of the firm is given by

$$U(u,P,R) = \pi(u) - D(P) - f(P-R), \tag{8}$$

---

1) It might appear strange to take into consideration a pollution level for an individual firm. However, P could be the current amount of slag in a foundry or the imissions of a firm into a lake which is used only by this firm; see, e.g., Sethi (1977).

where $\pi(u)$ denotes the net profit of the firm producing at rate u, D(P) is the disutility induced by the pollution level P, and f(P-R) is a penalty/reward function depending on the excess of actual and normal pollution level. The following assumptions are reasonable

$$\pi(u) > 0,\ \pi'(u) > 0,\ \pi''(u) < 0 \quad \text{for} \quad u > 0 \tag{9}$$

$$D(P) > 0,\ D'(P) > 0,\ D''(P) > 0 \quad \text{for} \quad P > 0. \tag{10}$$

In the sequel we restrict ourselves to *linear* penalty functions where the marginal penalty (purchase price of certificates) is *lower* than the marginal reward (selling price):

$$f(P-R) = \begin{cases} \alpha(P-R) & \text{for} \quad P > R \\ \beta(P-R) & \text{for} \quad P < R \end{cases} \tag{11}$$

with

$$0 < \alpha < \beta \tag{12}$$

Thus, if the actual pollution exceeds the normal level, the firm is penalized increasingly in accordance to the difference between both levels. On the other hand, the firm is rewarded proportionally to R-P provided this difference is positive. According to the incentive scheme (11), (12) set e.g. by the government, a market for certificates arises. Especially firms which are so small that devices for environmental protection (e.g. filters) can not be installed, will still profitably pollute, and *purchase* emission certificates to cover the excess P-R. For P < R, the firm can sell some of its certificates. Assumption (12) means that the planning authority sets a higher selling price to induce environmental protection.

Thus, the representative utility-maximizing firm is faced with the following infinite-time autonomous optimal control problem:

$$\max_{u} \int_0^\infty e^{-\rho t} U(u,P,R)\,dt \tag{13a}$$

$$\dot{P} = Z(u) - \delta P,\ P(0) = P_0 \tag{13b}$$

$$\dot{R} = m(P-R), \quad R(0) = R_0 \tag{13c}$$

$$0 \leq u \leq \bar{u} \tag{13d}$$

$$P \geq 0. \tag{13e}$$

There are two state variables, P and R, and one control, u. Both the initial actual pollution level and the normal pollution level are given. The path constraints (13de) are assumed to be automatically satisfied[1)].

---

1) For $\pi(0) = \infty$, $D'(0) = 0$, and $\bar{u}$ large enough no problems with the path restrictions arise.

3. ANALYSIS OF THE MODEL

Since the objective functional is not differentiable for P = R, (13) is not a standard optimal control problem. For its solution, a generalized maximum principle has to be used (see Clarke, 1983, Hartl and Sethi, 1984, Feichtinger and Hartl, 1985, 1986).

The current-value Hamiltonian is given by

$$H = \pi(u) - D(P) - f(P-R) + \lambda[Z(u) - \delta P] + \mu m(P-R),$$

where λ and μ are the adjoint variables corresponding to P and R, respectively.

The necessary optimality conditions are

$$u = \arg\max_{u} H, \text{ i.e. } \lambda = -\pi'(u)/Z'(u) \tag{15}$$

$$(\dot{\lambda}-\rho\lambda, \dot{\mu}-\rho\mu)\varepsilon -\partial_{(P,R)} H = (D'(P)+\delta\lambda-\mu m, \mu m) + \partial_{(P,R)} f(P-R), \tag{16}$$

where

$$\partial_{(P,R)} f(P-R) = \begin{cases} \{(\beta,-\beta)\} \\ \{(\gamma,-\gamma) \mid \alpha \leq \gamma \leq \beta\} \\ \{(\alpha,-\alpha)\} \end{cases} \quad \text{for} \quad P \begin{cases} < \\ = \\ > \end{cases} R. \tag{17}$$

Note that the right-hand side of the adjoint equation (16) is discontinuous for P = R. $\partial_{(P,R)} f(P-R)$ is the supergradient of f(P-R) with respect to (P,R) (see, e.g., Clarke, 1983).

To solve the four-dimensional canonical system of differential equations we specify the functions U_1, U_2 and Z as

$$\pi(u) = \sqrt{u}, \; D(P) = P^2, \; Z(u) = u^2 - \omega. \tag{18}$$

Moreover, the following values for the parameter values are chosen:

$$\rho = 0.1, \; \omega = 0.1, \; \delta = 0.1, \; m = 0.5, \; \alpha = 0.5, \; \beta = 1.0. \tag{19}$$

If we choose $\bar{u}$ large enough, e.g. $\bar{u} = 0.5$, then restriction (5) never becomes active.

Thus, the maximizing condition (15) yields

$$u = (-4\lambda)^{-2/3}.$$

Hence, the canonical system becomes

$$\left.\begin{aligned} \dot{P} &= (-4\lambda)^{-4/3} - \omega - 0.1\delta \\ \dot{R} &= 0.5(P-R) \\ \dot{\lambda} &= 0.2\lambda + 2P - 0.5\mu + \gamma \\ \dot{\mu} &= 0.6\mu - \gamma, \end{aligned}\right\} \tag{20}$$

where $\gamma = \left\{\begin{matrix}\beta\\ \alpha\end{matrix}\right\}$ for $P \left\{\begin{matrix}<\\ >\end{matrix}\right\} R$.

In order to solve the canonical system (20), the boundary value problem solver COLSYS was applied.

References for the collocation procedure COLSYS are Ascher et al. (1978), Steindl (1981); see also Steindl et al. (1986). By choosing appropriate cyclical boundary conditions, it turned out that a closed orbit exists. In the following section, we discuss the projections of this limit cycle into the (P,R) and (P,u) planes. Furthermore, the pollution and consumption trajectories are interpreted.

4. OPTIMAL OSCILLATIONS IN CONSUMPTION AND POLLUTION

Fig. 1 shows the limit cycle in the state space.

The cycle is followed counter-clockwise. In the points A_1 and A_3 it holds that $\dot{R} = 0$, whereas in A_2 and A_4 we have $\dot{P} = 0$. In A_1 the consumption rate reaches its maximum, whereas in A_3 it is minimal.

It seems convenient to distinguish four regimes in the cycle:

Regime 1: $P > R$, $\dot{R} > 0$, $\dot{P} > 0$, $\dot{u} < 0$. Starting in A_1 with low pollution and $P = R$ it is optimal to produce at high (maximal) but *decreasing* intensity. Due to the high production level, both the actual and the normal pollution increases, and penalty costs $\alpha(P-R)$ are charged. This phase ends with maximal actual pollution.

Regime 2: $P > R$, $\dot{R} > 0$, $\dot{P} < 0$, $\dot{u} < 0$. Since the actual pollution is at a high level, the production still has to decrease. Due to the low production rate the actual pollution decreases. The normal pollution, however, still increases. The regime ends as soon as $P = R$. Here the production intensity becomes minimal.

Regime 3: $P < R$, $\dot{R} < 0$, $\dot{P} < 0$, $\dot{u} > 0$. Since the firm produces at low level, both stocks of pollution decrease. It is optimal to *increase* the production rate. A bonus $\beta(R-P)$ is obtained. This phase ends with minimal actual pollution.

Regime 4: $P < R$, $\dot{R} < 0$, $\dot{P} > 0$, $\dot{u} > 0$. Since the actual pollution is minimal, it is still optimal to increase. As result, P increases again. However, R decreases util the end of this period. As soon as $P = R$ this regime ends and the cycle starts again with regime 1.

The corresponding limit cycle in the (P,C)-phase plane is sketched in Fig. 2. It is followed clockwise.

Figs. 3, 4 and 5 show the corresponding time paths for consumption, actual and normal pollution, respectively.

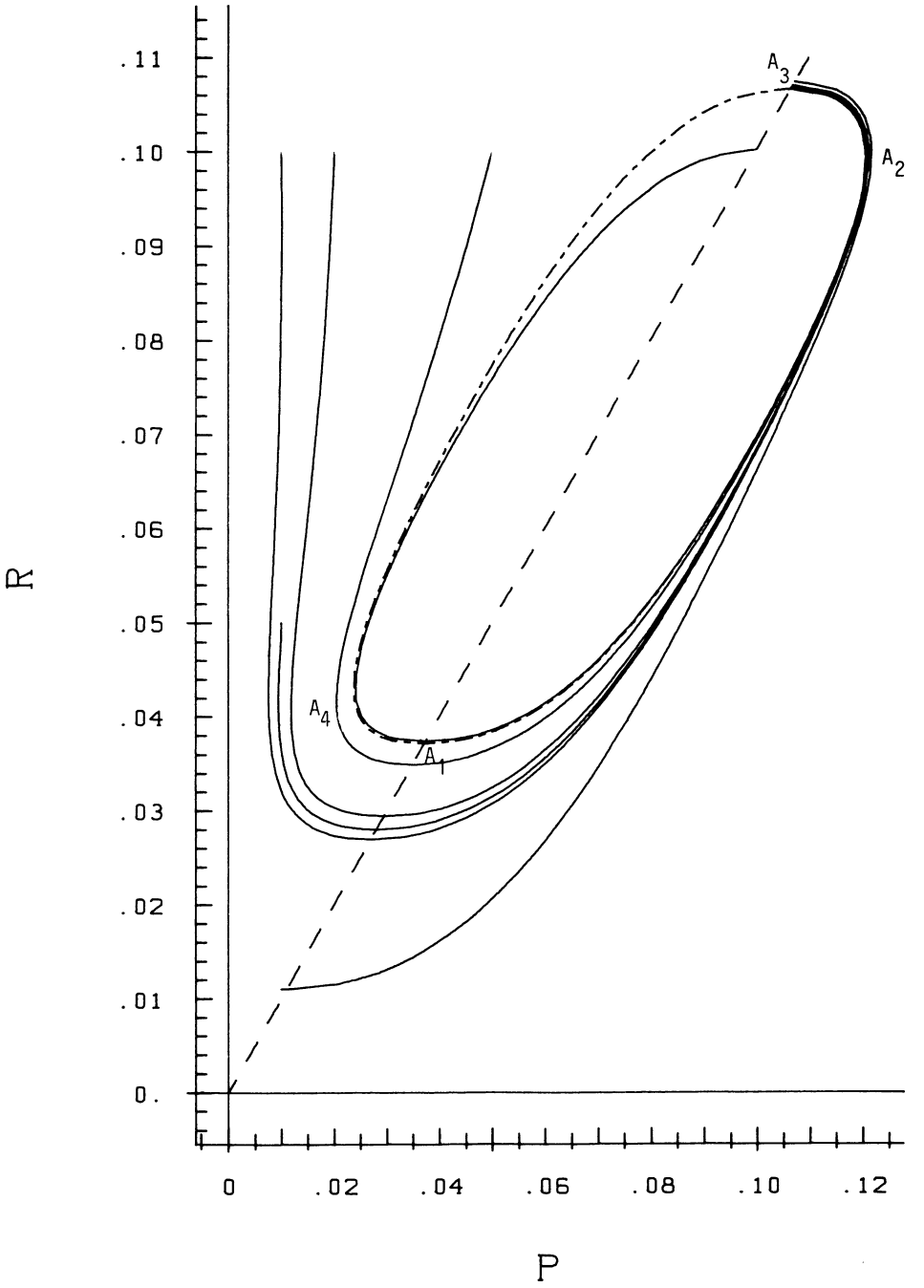


Fig. 1: Projection of the limit cycle in the state space

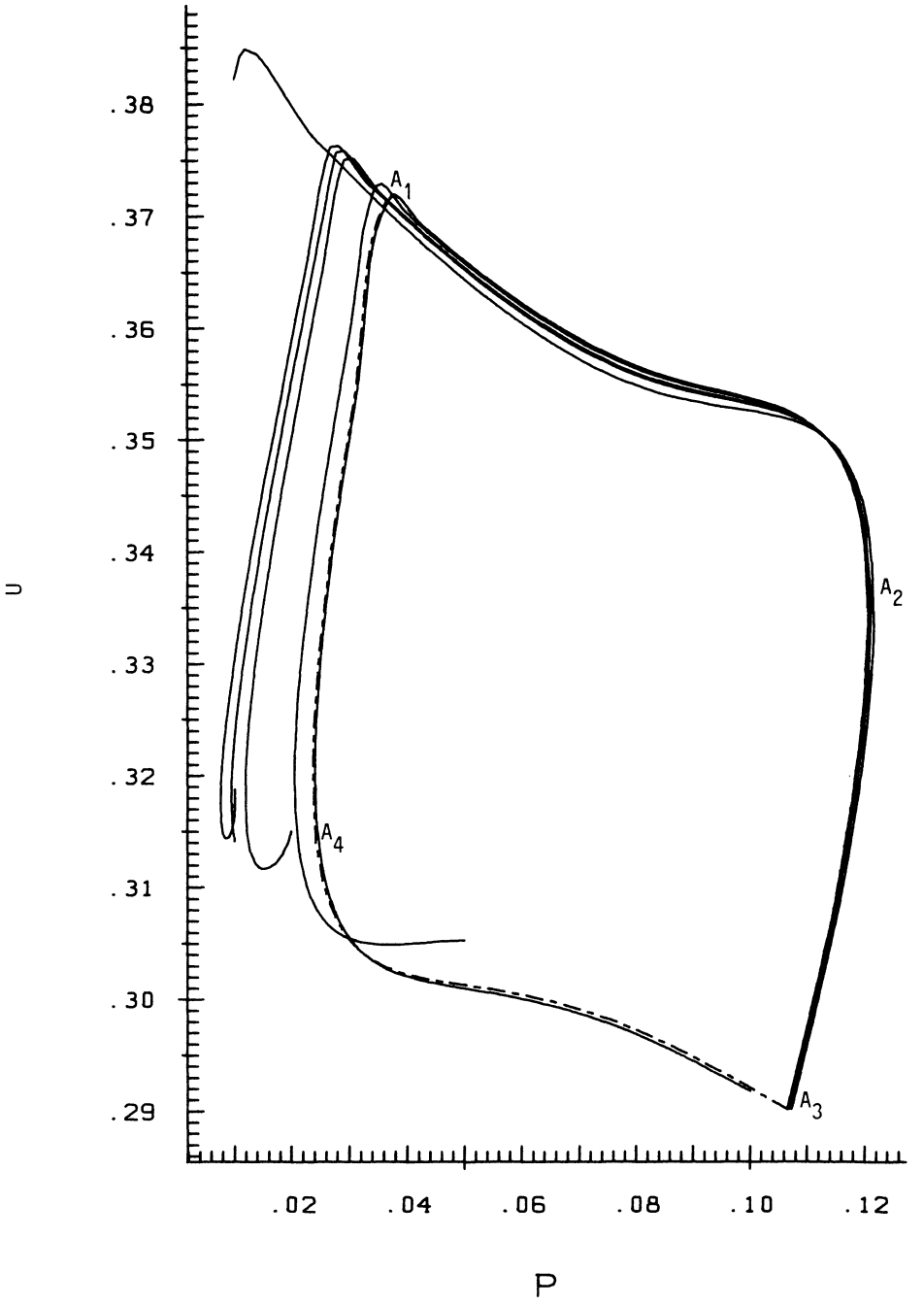


Fig. 2: Projection of the limit cycle in the phase plane of actual pollution and production rate

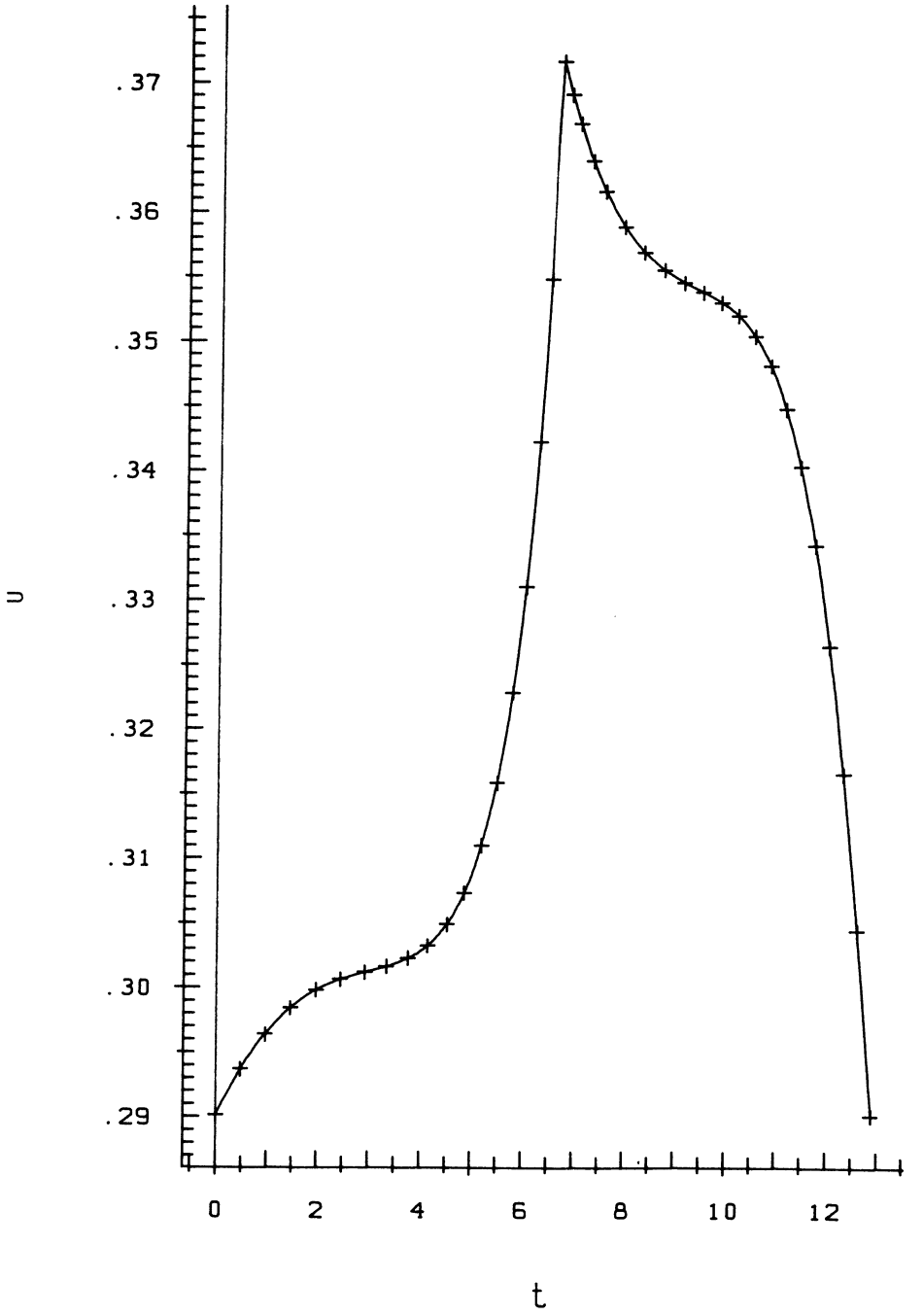


Fig. 3: Time path of the production rate

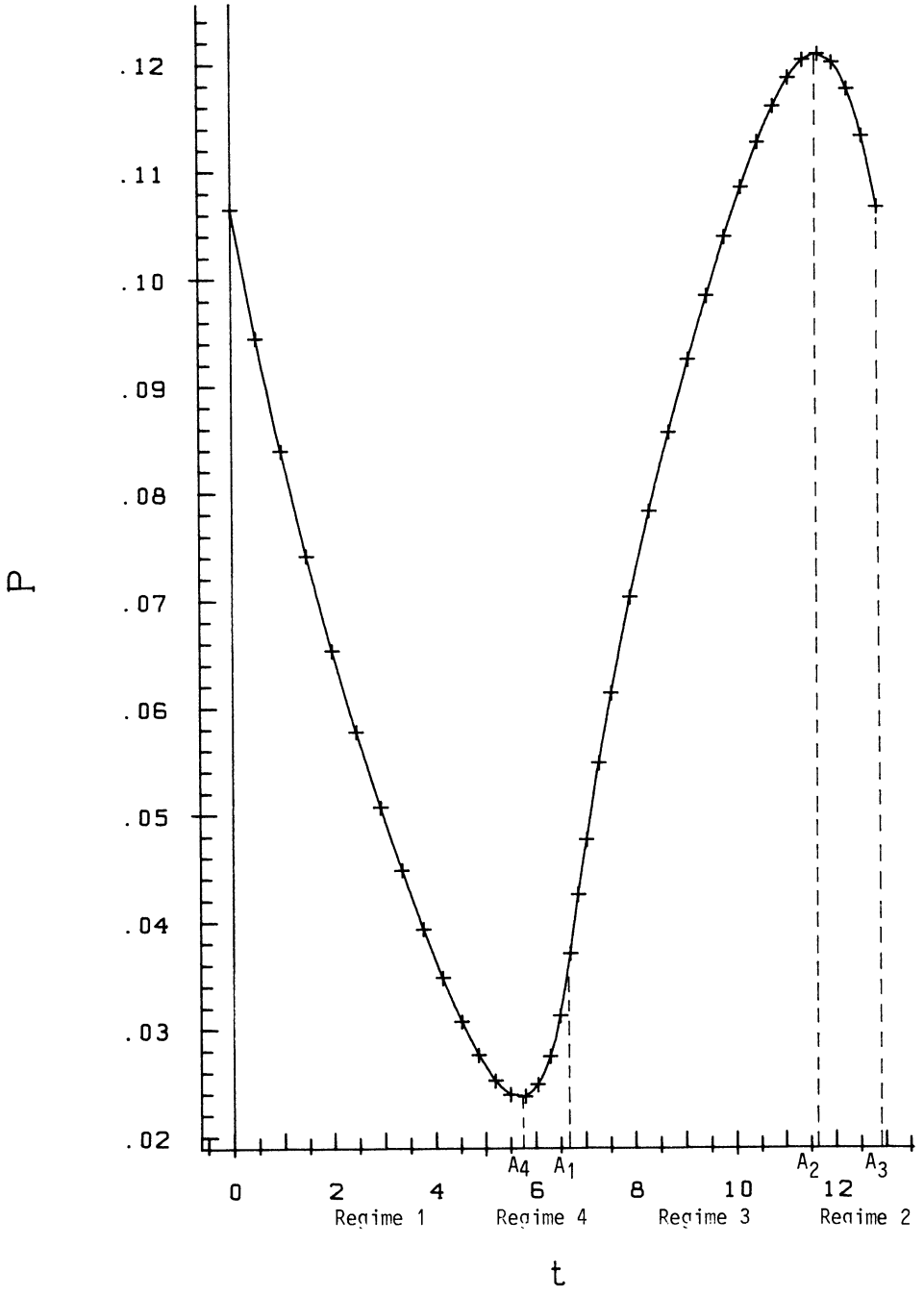


Fig. 4: Time path of the level of actual pollution

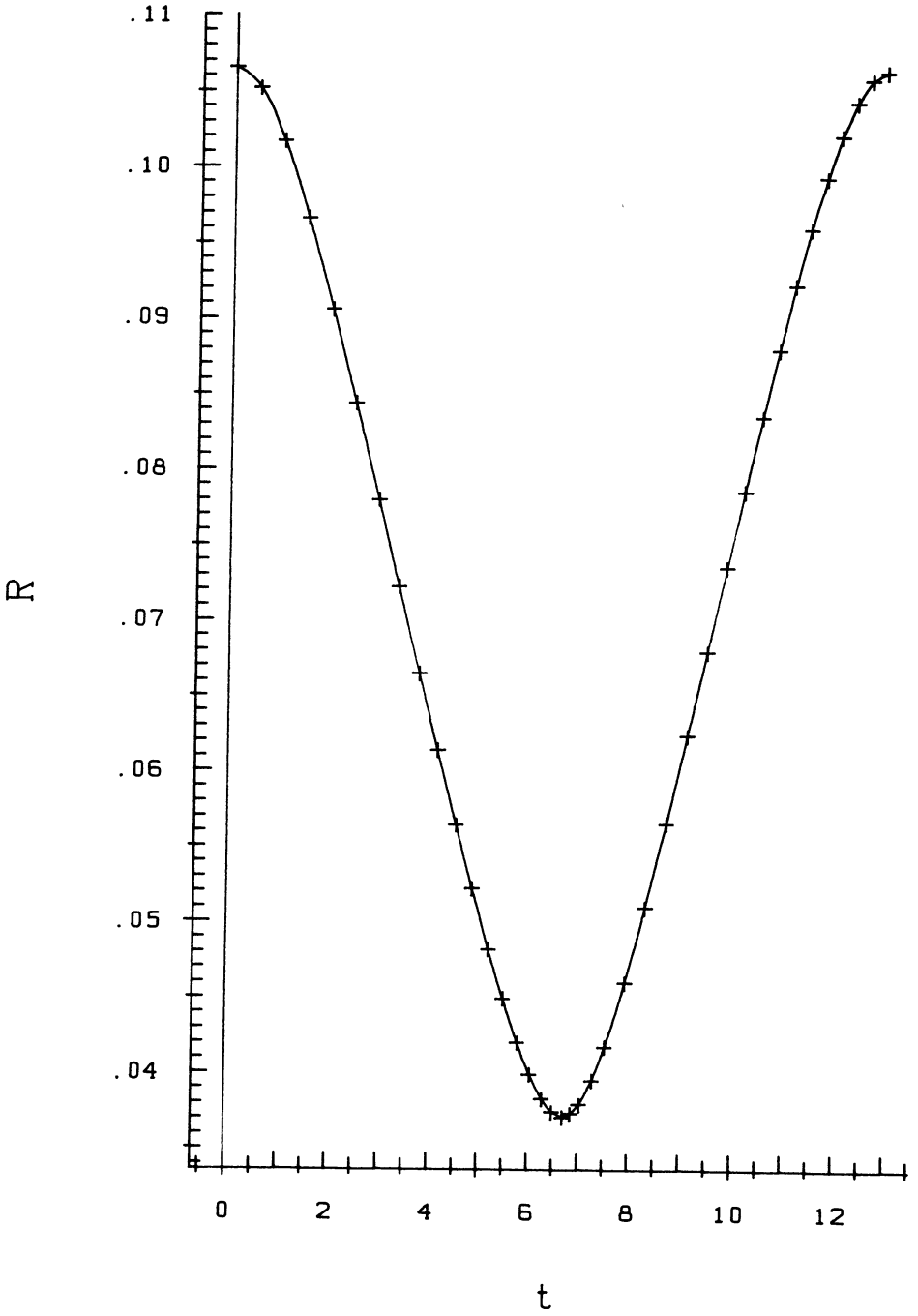


Fig. 5: Time path of the level of normal pollution

Some general insights can be obtained from these diagrams.

First, it is true that the normal stock of pollution always lags behind the actual level (see also (1)). Second, and more important, R(t) and u(t) show opposite behaviour (see Figs. 3 and 5). Third, Fig. 3 reveals that there are essentially two production levels, a high rate and a low one: The first prevails during regime 1, whereas in regime 3 the firm produces at a low rate. Both regimes 2 and 4 are switching phases, which are of short duration compared with regimes 1 and 3. Finally, the firm's optimal policy under the given incentive scheme is to produce at a high rate if the pollution is low. The increase in pollution due to this behaviour should be counteracted by a reduced production.

A two-dimensional manifold exists in the (P,R,λ,μ) space containing those paths that converge towards the limit cycle. In Figs. 1 and 2, five trajectories converging to the limit cycle are depicted. Apparently the convergence is very fast. After one or two periods, the paths almost coincide with the closed orbit.

5. CONCLUDING REMARKS

There are two main cinclusions. First, we have seen that there is an incentive scheme which is responsible for business cycles. Thus, the effect of a certain institutional arrangement which aims at the protection of the environment may run counter to its good intention[1)]. A useful incentive scheme should not generate cycles of the pollution level, instead it should be used to clean up the environment at a reasonably economic level.

Secondly, and more important than this economic conclusion, we have given a numerical proof of the existence and stability of a limit cycle in a pollution control model. There are a few papers in economic literature where the Hopf bifurcation technique is applied to prove the existence of limit cycles analytically (see Benhabib and Nishimura, 1979, Medio, 1986, Steindl et al., 1986, Feichtinger and Sorger, 1986). However, a complete analysis of this kind is very cumbersome. Not only that Hopf's approach gives no general information on the shape of the cycle. This justifies in our opinion the use of a numerical analysis. Furthermore, we obtained insight into the interdependencies of the model variables responsible for the oscillatory behaviour. A reasonable economic interpretation of the four regimes has been given.

Following Forster (1977), we have restricted ourselves to the investigation of the pure consumption-pollution trade-off. Clearly, more realistic models of environment control should include capital accumulation (see, e.g., Luptacik and Schubert, 1982 ab). Further research effort will concentrate on cyclical investment strategies in productive and abatement capital.

---

1) This behaviour is similar, e.g., to a result of Long and Siebert (1985).

REFERENCES

U. Ascher, J. Christiansen and R.D. Russell, "A collocation solver for mixed order systems of boundary value problems", Mathematics of Computation 33, 659-679 (1978).

W.J. Baumol and W.E. Oates, The Theory of Environmental Policy, Prentice-Hall, Englewood Cliffs, N.J. (1975).

J. Benhabib and K. Nishimura, "The Hopf bifurcation and the existence and stability of closed orbits in multisector models of optimal economic growth", J. Econ. Theory 21, 421-444 (1979).

F.H. Clarke, Optimization and Nonsmooth Analysis, Wiley, New York (1983).

G. Feichtinger and R.F. Hartl, "On the use of Hamiltonian and maximized Hamiltonian in nondifferentiable control theory", J. Optimiz. Theory Appl. 46, 493-504 (1985).

G. Feichtinger and R.F. Hartl, Optimale Kontrolle ökonomischer Prozesse: Anwendungen des Maximumprinzips in den Wirtschaftswissenschaften, de Gruyter, Berlin (1986).

G. Feichtinger and M. Luptacik, "Optimal production and abatement policies of a firm", Europ. J. Operational Res. 29, 274-285 (1987).

G. Feichtinger and G. Sorger, "Optimal oscillations in control models: how can constant demand lead to cyclical production?", Operations Res. Letters 5, 277-281 (1986).

B.A. Forster, "On a one state variable optimal control problem", in: Pitchford, S.J. Turnovsky (Eds.) Applications of Control Theory to Economic Analysis, North-Holland, Amsterdam, 35-56 (1977).

R.F. Hartl and S.P. Sethi, "Optimal control problems with differential inclusions: sufficiency conditions and an application to a production-inventory model", Optimal Control Applications and Methods 5, 289-307 (1984).

N.V. Long and H. Siebert, "Lay-off restraints, employment subsidies, and the demand for labour", in: G. Feichtinger (Ed.) Optimal Control Theory and Economic Analysis 2, North-Holland, Amsterdam, 293-312 (1985).

A. Luhmer, A. Steindl, G. Feichtinger, R.F. Hartl and G. Sorger, "ADPULS in continuous time", to appear in Europ. J. Operational Res. (1987).

M. Luptacik and U. Schubert, "Optimal Economic Growth and the Environment", in: W. Eichhorn et al. (Eds.) Economic Theory of Natural Resources, Physica, Würzburg, 455-468 (1982a).

M. Luptacik and U. Schubert, "Optimal investment policy in productive capacity and pollution abatement processes in a growing economy, in: G. Feichtinger (Ed.) Optimal Control Theory and Economic Analysis, North-Holland, Amsterdam, 231-243 (1982b).

A. Medio, "Oscillations in optimal growth models", Working paper, Dept. of Economics, Univ. of Venice (1986).

H.E. Ryder, Jr., and G.M. Heal, "Optimal growth with intertemporally dependent preferences", Rev. Economic Studies 121, 1-31 (1973).

J.R. Schmalensee, The Economics of Advertising, North-Holland, Amsterdam (1972).

S.P. Sethi, "A linear bang-bang model of firm behavior and water quality", IEEE Trans. Autom. Contr. AC-21, 706-714 (1977).

H. Siebert, "Ökonomische Anreize in der TA-Luft?", Wirtschaftsdienst 62, 560-564 (1982).

A. Steindl, "COLSYS: Ein Kollokationsverfahren zur Lösung von Randwertproblemen bei Systemen gewöhnlicher Differentialgleichungen", Diplomarbeit, Techn. Univ. Wien (1981).

A. Steindl, G. Feichtinger, R.F. Hartl and G. Sorger, "On the optimality of cyclical employment policies: a numerical investigation", J. of Economic Dynamics and Control 10, 457-466 (1986).

L. Wicke, Umweltökonomie: Eine praxisorientierte Einführung, F. Vahlen, München (1982).

LOCATION PROBLEM AND ITS APPLICATIONS IN DISTRIBUTION OF PETROL PRODUCTS

B.Mażbic-Kulma, E.Komorowska, J.Stępień

A strategic issue which concerns investment planners is where to best locate new warehouses, factories or intermediate stocking points. In this paper we consider the location problem of bulk warehouses for petrol products in a region. The problem is formulated as a capacitated facility location problem.

Computational results for a set of real-size problems are finally discussed.

1. INTRODUCTION

The problem of locating warehouses is not new to operations research and is known from a rich literature. In this paper we consider the location problem of bulk warehouses for petrol products in a region. It may happen that transport costs are compared to investment costs. Therefore it is very interesting where to locate new warehouses in order to minimize total investment and transport costs. This problem may be considered in the framework of designing the efficient distribution network configurations. The Desing Office of the Petrol Industry was interested in the application of this problem and its solutions to a distribution network for petrol products.

Location models can be clossified according to:

1.Whether the underlying distribution network /arcs and /or nodes / is capacitated or uncapacitated.
2.The number of warehous echelons or levels /zero,single or multiple/,
3.The number of commodities /single or multiple/,
4.The underlying cost structure for arcs and/or nodes /linear or nonlinear/,
5.Whether the planning horizon is static or dynamic,
6.Patterns of demand /e.g. deterministic or stochastic, influence of location ,etc./,
7.The ability to accomodate side constraints.

The problem formulations range form single-commodity linear deterministic models to multicommodity non-linear stochastic models.

2. MATHEMATICAL MODEL

Our problem of warehouses location for petrol products distribution network is as follows:

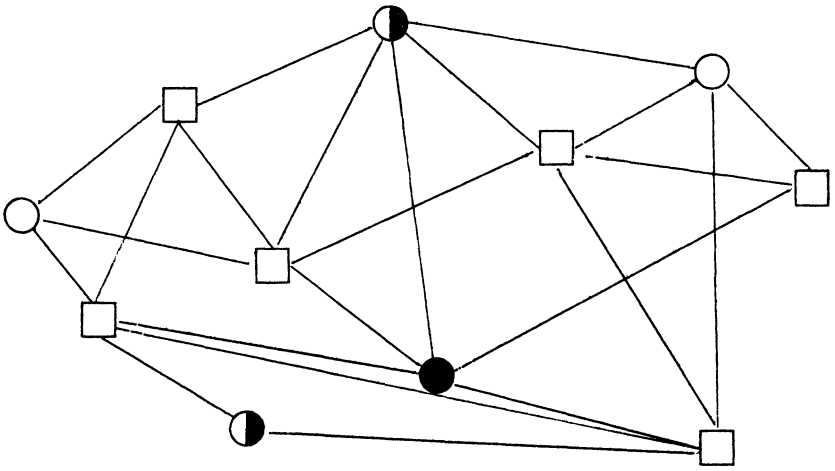


Fig.1.

□ - destinations of petrol products,

●◑○ - warehouses of petrol products.

We have three kinds of warehouses of petrol products. The first group contains existing warehouses which have fixed capacity. The second group contains existing warehouses too, but there capacity may be extended. The third group contains the feasible location places of new warehouses which will be build in future.

Our problem may be formulated as follows:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (t_{ij}+s_i)x_{ij} + \sum_{i=1}^{m} k_i \operatorname{sgn}\left(\sum_{j=1}^{n} x_{ij}\right) \qquad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{m} x_{ij} = b_j \qquad (j=1,2,\ldots,n) \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i \qquad (i=1,2,\ldots,m) \qquad (3)$$

$$x_{ij} \geq 0 \quad (i=1,2,\ldots,m) \ , \ (j=1,2,\ldots,n) \qquad (4)$$

Where:

x_{ij} - the number of units of customer j's demand satisfied by facility i.

$y_i=1$ if facility i is established, 0 otherwise

c_{ij} - the unit investment and distribution costs associated with satisfying customer j's demand from facility i ($c_{ij} = t_{ij} + s_i$)

k_i - the fixed costs of facility i

b_j - the demand of customer j

a_i - the capacity of facility i

m - the number of potential warehouse's location

n - the number of customers.

This problem may be solved by the method of succesive calculations |7| , |24| .

Let $M = \{1,2,\ldots,m\}$ denote the number of possible locations of warehouses. Let Ω ,where $\omega \in \Omega$, denote the set of all subsets of the set M. A subset ω will be called a variant of locations.

Of course the number of all subsets of the set M is 2^m , i.e. the maximal number of possible solutions is 2^m too.

Let F be a function defined on the set Ω . In order to reduce the number of combinations of plant location to study, we can use the following property and theorem.

Cherenin Property

For arbitrary $\omega_1, \omega_2 \subset M$,

$$F(\omega_1) + F(\omega_2) - F(\omega_1 \cup \omega_2) - F(\omega_1 \cap \omega_2) \leq 0 \qquad (5)$$

Cherenin Theorem

Let a function $F(\omega)$ possess Cherenin Property and let ω^* be an optimal subset. Then, for any finite sequence $\{\omega_i\}$ containing ω^* , where $\omega_i \subset \omega_{i+1}$,the function $F(\omega)$ is monotone nonincreasing to ω^* and monotone nondecreasing after ω^*.

The function wich posses this theorem is called submodular. It is proved by Babayev that Cherenin Theorem and properties formulated by Frieze for submodular functions are equivalent.

The quality of a location variant may be evaluated by the following function $F(\omega)$ defined on the set Ω.

$$F(\omega) = \sum_{i \in \omega} k_i + L(\omega) \tag{6}$$

where:

$$L(\omega) = \min \sum_{i \in \omega} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{7}$$

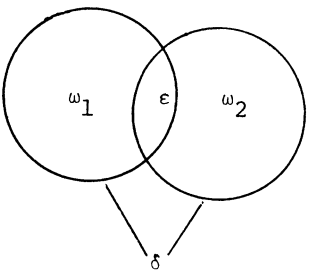
$$\text{subject to} \quad \sum_{i \in \omega} x_{ij} \geqslant b_j \quad (j=1,2,\ldots,n) \tag{8}$$

$$\sum_{j=1}^{n} x_{ij} \leqslant a_i \quad i \in \omega \tag{9}$$

$$x_{ij} \geqslant 0 \quad , \; i \in \omega, \; (j=1,2,\ldots,n) \tag{10}$$

It may be proved, that this function possess Cherenin Property.

$$F(\omega_1) + F(\omega_2) - F(\omega_1 \cup \omega_2) - F(\omega_1 \cap \omega_2) \leqslant 0$$



$\varepsilon = \omega_1 \cap \omega_2$

$\delta = \omega_1 \cup \omega_2$

Fig.2.

$$F(\omega_1) + F(\omega_2) - F(\delta) - F(\varepsilon) = \underbrace{\sum_{i\in\omega_1} k_i + \sum_{i\in\omega_2} k_i - \sum_{i\in\delta} k_i - \sum_{i\in\varepsilon} k_i}_{=0} +$$

$$+ \min_{i\in\omega_1} \sum_{j=1}^{n} c_{ij}x_{ij} + \min_{i\in\omega_2} \sum_{j=1}^{n} c_{ij}x_{ij} - \min_{i\in\delta} \sum_{j=1}^{n} c_{ij}x_{ij} -$$

$$- \min_{i\in\varepsilon} \sum_{j=1}^{n} c_{ij}x_{ij} \leq 0 \qquad (11)$$

$$\min_{i\in\delta} \sum_{j=1}^{n} c_{ij}x_{ij} = \min_{i\in\omega_1} \sum_{j=1}^{n} c_{ij}x_{ij} \quad \text{or} \quad \min_{i\in\delta} \sum_{j=1}^{n} c_{ij}x_{ij} = \min_{i\in\omega_2} \sum_{j=1}^{n} c_{ij}x_{ij} \qquad (12)$$

$$\min_{i\in\varepsilon} \sum_{j=1}^{n} c_{ij}x_{ij} \geq \min_{i\in\omega_1} \sum_{j=1}^{n} c_{ij}x_{ij} \quad \text{and} \quad \min_{i\in\varepsilon} \sum_{j=1}^{n} c_{ij}x_{ij} \geq \min_{i\in\omega_2} \sum_{j=1}^{n} c_{ij}x_{ij} \qquad (13)$$

3. METHOD OF SUCCESIVE CALCULATIONS

The capacitated facility location problem described by (1) - (4) may be solved by means of an algorithm whose fundamental points are as follows:

- Each vertex of the search tree corresponds to a subset ω. Furthermore, all subsets having the same number of elements create one level of the graph. The vertices at the same level are created in lexicographical order. Let $|\omega| = k$ be called the index of the level, and level k be denoted as Ω_k. The search tree is created level by level from k=0 to k=m.

- From the vertex ω , $|\omega| = k$ we may generate m-k vertices ω_s from the level k+1 in the following way:
  $\omega_s = \omega \cup \{s\}$, where $s \in M \setminus \omega$.
  In the search tree for each vertex ω , $|\omega| = k$ there are (m-k) arcs towards to the next level and k arcs originating from the previous level. In Fig.2 the search tree for m=4 is shown.

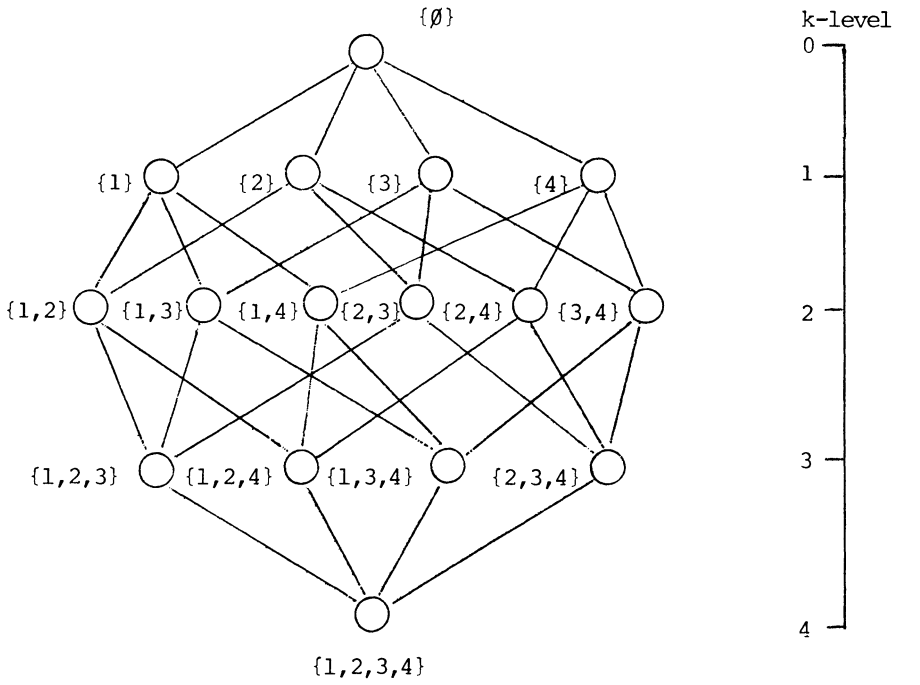


Fig.3.

- At a given level k only so-called perpective vertices are remembered. The vertex ω , $|\omega| = k$ is called unperpective if, for all α from the previous level such that $\alpha \subset \omega$, $f(\alpha) < f(\omega)$. The set of all the perspective vertices at level k is denoted R_k, $R_k \subset \Omega_k$.If the vertex ω is

unperspective, the associated transportation problem needs not to be solved. Otherwise this has to be done. Denote by $L(\omega)$ the corresponding transport costs. Consider a vertex ω with fixed cost $k(\omega) = \sum_{i \in \omega} k_i$.

The total costs associated with vertex ω are then

$$F(\omega) = k(\omega) + L(\omega)$$

Let $L(M)$ denote the optimum solution of the transport problem when all warehouses are open (corresponding to the lowest value for the transport costs). Let F^* be the current best solution.

Then $k(\omega) + L(\omega) > F^*$ implies that ω is unperspective.

- The search tree is generated level by level, but at the given level k we generate and try out only these vertices of the previous level.

The basic steps of the algorithm proceed as follows:

step 0 Calculate $L(M)$

$F^* = \infty$, $\omega^* = \{\emptyset\}$

$\Omega_k = \{\emptyset\}$ for $k = \emptyset,1,2,\ldots,m$

$R_k = \{\emptyset\}$ for $k = 1,2,\ldots,m$

$k = \emptyset$

step 1 $k = k+1$

Generate all subsets $\omega_s \in \Omega_k^*$ of the k-th level from the subsets $\omega \in R_{k-1}$. R_{k-1} is the set of perspective vertices of the previous level.

Elements of $\Omega_k^* \subset \Omega_k$ must be arranged in lexicographical order.

step 2 For each $\omega \in \Omega_k^*$

a/ if $k(\omega) + L(M) < F^*$, go to step 2b, otherwise ω is unperspective, examine next ω from Ω_k

b/ calculate $F(\omega)$. This needs solving of associated transportation problem

if $F(\omega) > F(\alpha)$ for all α from level $k-1, \alpha$ ω, then ω is unperspective, next ω from Ω_k^* must be examined, otherwise go to step 2c

c/ Calculate $F(\omega)$. This needs solving of associated transportation problem.

if $F(\omega) > F(\alpha)$ for all α , $\alpha \in \Omega_{k-1}, \alpha \subset \omega$, then ω is unperspective, next ω from Ω_k^* must be examined. Otherwise go to step 2d.

d/ if $F(\omega) < F^*$, $F^* = F(\omega)$, $\omega^* = \omega$

$R_k = R_k \quad \{\omega\}$

step 3 if $R_k \neq 0$, go to step 1

Otherwise algorithm is finished ω^* is the optimal solution and the optimal value of the objective function eaquals F^*.

The algorithm is ended , when we have only unperspective veitices.

4. EXAMPLE

The problem has been tested on a warehouse location problem related to the Design Office of Petrol Industry for petrol products distribution in some regions of Poland. The maximal size of real problem treated was 20 warehouses and 40 customers.

The demands and warehouse capacities for one region are indicated in Tables 1 and 2. The unit transportation costs c_{ij} have been assumed to be proportional to the road distances (measured in kilometers).

Table 1 , Data for a test problem: customer demands

No	Customer	Demand (10^3 tons)	No	Customer	Demand (10^3 tons)
1	Działdowo	5.6	18	Garwolin	5.1
2	Ciechanów	5.6	19	Mińsk Maz.	15.3
3	Mława	3.9	20	Siedlce	6.8
4	Płońsk	2.8	21	Sokołów Podl.	4.6
5	Pułtusk	3.1	22	Węgrów	8.6
6	Zuromin	2.0	23	Zelechów	4.6
7	Nasielsk	2.8	24	Modry	4.0
8	Raciąż	2.2	25	Stoczek Łuk.	4.0
9	Maków Maz.	8.4	26	Grodzisk Maz.	12.8
10	Ostrołęcka	15.2	27	Nowy Dwór Maz.	12.8
11	Przasnysz	7.6	28	Otwock	8.6
12	Ostrów Maz.	12.9	29	Piaseczno	12.8
13	Wyszków	10.6	30	Pruszków	55.6
14	Myszyniec	6.1	31	Wołomin	8.6
15	Rożan	8.4	32	Warszawa	295.4
16	Chorzele	6.8	33	Błonie	12.8
17	Łuków	4.0	34	Góra Kalwaria	8.6

Table 2. Data for a test problem: warehouse capacities

No	Warehouse	Maximal capacity (10^3 tons)	No	Warehouse	Maximal capacity (10^3 tons)	Investment Costs
1	Praga	24.9	8	Emilianów	236.3	-
2	Sokołów	1.7	9	Ostrołęka	600.0	2000
3	Mława	0.0	10	Przasnycz	600.0	2000
4	Ciechanów	5.6	11	Maków Maz.	600.0	2000
5	Działdowo	3.1	12	Siedlce	600.0	2000
6	Mościska	200.0 (600.0)	13	Zelechów	600.0	2000
7	Ostrów Maz.	1.2	14	Łuków	600.0	2000

Now, we would like to present how this algorithm works, (look at the Fig.4).

It easy to check, that the optimal combination is {11,12,15} with F = 19434 at the third level. At the next level we dont have perspective vertices.

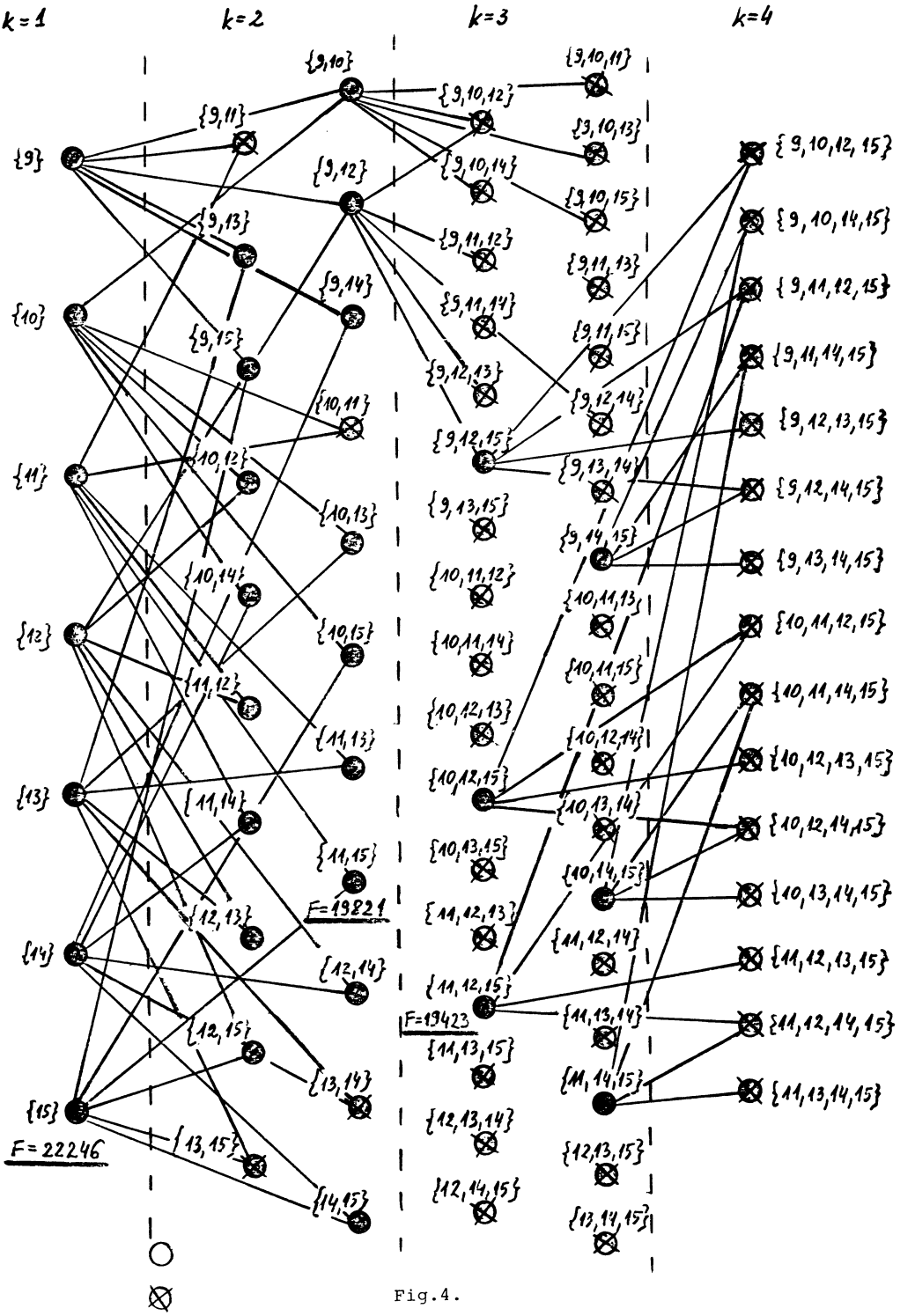

k=1
k=2
k=3
k=4
{9}
{10}
{11}
{12}
{13}
{14}
{15}
{9,10}
{9,11}
{9,12}
{9,13}
{9,14}
{9,15}
{10,11}
{10,12}
{10,13}
{10,14}
{10,15}
{11,12}
{11,13}
{11,14}
{11,15}
{12,13}
{12,14}
{12,15}
{13,14}
{13,15}
{14,15}
F=22246
F=19821
{9,10,11}
{9,10,12}
{9,10,13}
{9,10,14}
{9,10,15}
{9,11,12}
{9,11,13}
{9,11,14}
{9,11,15}
{9,12,13}
{9,12,14}
{9,12,15}
{9,13,14}
{9,13,15}
{9,14,15}
{10,11,12}
{10,11,13}
{10,11,14}
{10,11,15}
{10,12,13}
{10,12,14}
{10,12,15}
{10,13,14}
{10,13,15}
{10,14,15}
{11,12,13}
{11,12,14}
{11,12,15}
{11,13,14}
{11,13,15}
{11,14,15}
{12,13,14}
{12,13,15}
{12,14,15}
{13,14,15}
F=19423
{9,10,12,15}
{9,10,14,15}
{9,11,12,15}
{9,11,14,15}
{9,12,13,15}
{9,12,14,15}
{9,13,14,15}
{10,11,12,15}
{10,11,14,15}
{10,12,13,15}
{10,12,14,15}
{10,13,14,15}
{11,12,13,15}
{11,12,14,15}
{11,13,14,15}

Fig.4.

5. CONCLUSIONS

The algorithm proposed in this paper was programmed in Fortran for the IBM PC. This program is used in "NAFTOPROJEKT" Desing Office of Petrol Products Industry for solving real problems in investment planning on petrol products distribution network.

In further we plan to apply the multicomodity location model and dynamic location model for solving the milk products distribution problem in Rzeszów province in our country.

REFERENCES

|1| Aikens C.H. "Facility location models for distribution planning" EJOR 22, 1985, pp.263-279.

|2| Akinc U., Khumawala B.M. "An efficient branch and bound algorithm for the capacitated warehouses location problem" Man. Sci. , vol. 23, No 6, 1977, pp.585-594.

|3| Alcouffe A., Muratet G. "Optimal location of plants" Man. Sci., vol.23, No 3, 1976 , pp.267-274.

|4| Babayev D.A. "Comments on the note of Frieze" Math. Progr. 7, 1974 , pp.248-252.

|5| Bartezzaghi E., Coloroni A., Palermo P.C. " A search tree algorithm for plant location problems" EJOR, vol 7. No 4 , 1981, pp,371-279.

|6| Bilde O., Krarup J. "Sharp lower bounds for the simple location problem" Anuals of Discrete Math. 1, 1977, pp.79 - 97.

|7| Cherenin V.P., Chaschaturov V.R. "Solving by the method of succesive calculations a class of plant location problems" (in Russian). Economics and Mathematical Methods 2, Science , Moscow.

|8| Christofides N., Beasley J.E. "Externsions to a Lagrangean relaxation approach for the capacitated warehouse Location problem" . EJOR, vol 12, 1983,pp. 19-28.

|9| Dearing P.M., Newruck F.C. "A capacitated bottleneck facility location problem". Man.Sci.,No 11,1979, pp.1093-1104.

|10| Elson D.G. "Site location via mixed-integer programming" Oper. Res. Quart, No 23 , 1972 , pp.31 - 43.

|11| Erlenkotter 0. "A dual-based procedure for uncapacitated facility location" Op. Res.,No 26, 1978 , pp.992-1009.

|12| Frieze A.A. "A Cost Function Property for Plant Location Problem" . Math. Progr. No 7, 1974, pp. 244 -248.

|13| Garfinkel R.S., Rao M.R. "The bottleneck transportation problem" Naval. Res. Log. Quart. No 18, 1971,pp. 464-472.

|14| Geoffrion A.M., Graves G.W. "Multicommodity distribution system design by Benders decomposition". Man. Sci., No 20, 1974, pp. 822- 844.

|15| Geoffrion A.M., McBride R. "Lagrangean relaxation applied to the capacitated facility location problem".AIIE Transactions No 10, 1978, pp. 40 - 47.

|16| Grishukhin V.P. "Submodular problems and algorithms for solving them" in Russian. USSR Ac. of Sciences J. Tekhnicheskaia Kibernetika, No 1, 1983,pp. 188 - 195.

|17| Hansen P., Thiesse J.F., Hanjoul P. "Simple plant location under uniform delivered pricing" EJOR, vol.6,1981, pp. 94-103.

|18| Karhazis J.,Boffey T.B. "The multicommodity facilities location problem". Journal of Op. Res.Soc.,No 32,1981, pp.803 - 814.

|19| Kaufman L., Eese M.V., Hansen P. "A plant and warehouse location problem" Oper.Res. Quart.No 28, 1977, pp. 547 - 554.

|20| Khumawala B.M., "An efficient branch-and-bound algorithm for the warehouse location problem" Man. Sci.No 18,1972, B718 - B731.

|21| Khumawala B.M.,Neebe A.W., "A note of Warszawski's multicommodity location problem" Journal of Op.Res.Soc., No 6, 1978,pp.171 - 172.

|22| Kim.K.,"Location problem" CEMI works, Moscov 1984 in Russian.

|23| Le Blanc L.J., "A heuristic approach for large scale discrete stochastic transportation-location problems" Computers and Mathematics with Applications,No 3,1977, pp. 87 - 94 .

|24| Mielnikov D.I. "The method of succesive calculations and the plant location problem with nonfixed demands" in work on discrete mathematics, 1973, Moscov, "Science" publishers.

|25| Nauss R.M. "An improved algorithm for the capacitated facility location problem" Journal of Op.Res.Soc. No 29, 1978, pp. 1195 - 1201.

|26| Roodman G.M.,Schwartz L.B. "Optimal and heuristic facility phase-out strategies" AIIE Trans., No 7, 1975, pp.177 - 184.

|27| Spielberg K. "Plant location with generalized search origin". Man.Sci., No 16, 1969, pp.165 - 1978.

|28| Spielberg K. "Algorithms for the simple plant-location problem with some side conditions" Op. Res.,No 17,1969, pp. 85 - 111.

|29| Tcha D., Lee B. " A branch-and-bound algorithm for the multi-level uncapacitated facility location-problem" EJOR,No 18,1984, pp. 35 - 43.

|30| Trubin A.V. "Two classes of problems of location on tree-like nets" in Russian, Kibernetika,No 4, 1983.

|31| Van Roy T.J.,Erlenkotter D. "Dual based procedure for dynamic facility location" Man.Sci.No 28,1982, pp. 1091 - 1105.

|32| Warszawski A. "Multi-dimensional location problems" Op.Res.Quart.,No 24, 1973, pp.165-179.

|33| Wesolowski G.O., Truscott W.G., "The multiperiod location-allocation problem with relation of facilities" Man.Sci. ,No 22, 1975,pp.57-65.

Vol. 211: P. van den Heuvel, The Stability of a Macroeconomic System with Quantity Constraints. VII, 169 pages. 1983.

Vol. 212: R. Sato and T. Nôno, Invariance Principles and the Structure of Technology. V, 94 pages. 1983.

Vol. 213: Aspiration Levels in Bargaining and Economic Decision Making. Proceedings, 1982. Edited by R. Tietz. VIII, 406 pages. 1983.

Vol. 214: M. Faber, H. Niemes und G. Stephan, Entropie, Umweltschutz und Rohstoffverbrauch. IX, 181 Seiten. 1983.

Vol. 215: Semi-Infinite Programming and Applications. Proceedings, 1981. Edited by A. V. Fiacco and K. O. Kortanek. XI, 322 pages. 1983.

Vol. 216: H. H. Müller, Fiscal Policies in a General Equilibrium Model with Persistent Unemployment. VI, 92 pages. 1983.

Vol. 217: Ch. Grootaert, The Relation Between Final Demand and Income Distribution. XIV, 105 pages. 1983.

Vol. 218: P. van Loon, A Dynamic Theory of the Firm: Production, Finance and Investment. VII, 191 pages. 1983.

Vol. 219: E. van Damme, Refinements of the Nash Equilibrium Concept. VI, 151 pages. 1983.

Vol. 220: M. Aoki, Notes on Economic Time Series Analysis: System Theoretic Perspectives. IX, 249 pages. 1983.

Vol. 221: S. Nakamura, An Inter-Industry Translog Model of Prices and Technical Change for the West German Economy. XIV, 290 pages. 1984.

Vol. 222: P. Meier, Energy Systems Analysis for Developing Countries. VI, 344 pages. 1984.

Vol. 223: W. Trockel, Market Demand. VIII, 205 pages. 1984.

Vol. 224: M. Kiy, Ein disaggregiertes Prognosesystem für die Bundesrepublik Deutschland. XVIII, 276 Seiten. 1984.

Vol. 225: T. R. von Ungern-Sternberg, Zur Analyse von Märkten mit unvollständiger Nachfragerinformation. IX, 125 Seiten. 1984

Vol. 226: Selected Topics in Operations Research and Mathematical Economics. Proceedings, 1983. Edited by G. Hammer and D. Pallaschke. IX, 478 pages. 1984.

Vol. 227: Risk and Capital. Proceedings, 1983. Edited by G. Bamberg and K. Spremann. VII, 306 pages. 1984.

Vol. 228: Nonlinear Models of Fluctuating Growth. Proceedings, 1983. Edited by R. M. Goodwin, M. Krüger and A. Vercelli. XVII, 277 pages. 1984.

Vol. 229: Interactive Decision Analysis. Proceedings, 1983. Edited by M. Grauer and A. P. Wierzbicki. VIII, 269 pages. 1984.

Vol. 230: Macro-Economic Planning with Conflicting Goals. Proceedings, 1982. Edited by M. Despontin, P. Nijkamp and J. Spronk. VI, 297 pages. 1984.

Vol. 231: G. F. Newell, The M/M/∞ Service System with Ranked Servers in Heavy Traffic. XI, 126 pages. 1984.

Vol. 232: L. Bauwens, Bayesian Full Information Analysis of Simultaneous Equation Models Using Integration by Monte Carlo. VI, 114 pages. 1984.

Vol. 233: G. Wagenhals, The World Copper Market. XI, 190 pages. 1984.

Vol. 234: B. C. Eaves, A Course in Triangulations for Solving Equations with Deformations. III, 302 pages. 1984.

Vol. 235: Stochastic Models in Reliability Theory. Proceedings, 1984. Edited by S. Osaki and Y. Hatoyama. VII, 212 pages. 1984.

Vol. 236: G. Gandolfo, P. C. Padoan, A Disequilibrium Model of Real and Financial Accumulation in an Open Economy. VI, 172 pages. 1984.

Vol. 237: Misspecification Analysis. Proceedings, 1983. Edited by T. K. Dijkstra. V, 129 pages. 1984.

Vol. 238: W. Domschke, A. Drexl, Location and Layout Planning. IV, 134 pages. 1985.

Vol. 239: Microeconomic Models of Housing Markets. Edited by K. Stahl. VII, 197 pages. 1985.

Vol. 240: Contributions to Operations Research. Proceedings, 1984. Edited by K. Neumann and D. Pallaschke. V, 190 pages. 1985.

Vol. 241: U. Wittmann, Das Konzept rationaler Preiserwartungen. XI, 310 Seiten. 1985.

Vol. 242: Decision Making with Multiple Objectives. Proceedings, 1984. Edited by Y. Y. Haimes and V. Chankong. XI, 571 pages. 1985.

Vol. 243: Integer Programming and Related Areas. A Classified Bibliography 1981–1984. Edited by R. von Randow. XX, 386 pages. 1985.

Vol. 244: Advances in Equilibrium Theory. Proceedings, 1984. Edited by C. D. Aliprantis, O. Burkinshaw and N. J. Rothman. II, 235 pages. 1985.

Vol. 245: J. E. M. Wilhelm, Arbitrage Theory. VII, 114 pages. 1985.

Vol. 246: P. W. Otter, Dynamic Feature Space Modelling, Filtering and Self-Tuning Control of Stochastic Systems. XIV, 177 pages. 1985.

Vol. 247: Optimization and Discrete Choice in Urban Systems. Proceedings, 1983. Edited by B. G. Hutchinson, P. Nijkamp and M. Batty. VI, 371 pages. 1985.

Vol. 248: Plural Rationality and Interactive Decision Processes. Proceedings, 1984. Edited by M. Grauer, M. Thompson and A. P. Wierzbicki. VI, 354 pages. 1985.

Vol. 249: Spatial Price Equilibrium: Advances in Theory, Computation and Application. Proceedings, 1984. Edited by P. T. Harker. VII, 277 pages. 1985.

Vol. 250: M. Roubens, Ph. Vincke, Preference Modelling. VIII, 94 pages. 1985.

Vol. 251: Input-Output Modeling. Proceedings, 1984. Edited by A. Smyshlyaev. VI, 261 pages. 1985.

Vol. 252: A. Birolini, On the Use of Stochastic Processes in Modeling Reliability Problems. VI, 105 pages. 1985.

Vol. 253: C. Withagen, Economic Theory and International Trade in Natural Exhaustible Resources. VI, 172 pages. 1985.

Vol. 254: S. Müller, Arbitrage Pricing of Contingent Claims. VIII, 151 pages. 1985.

Vol. 255: Nondifferentiable Optimization: Motivations and Applications. Proceedings, 1984. Edited by V. F. Demyanov and D. Pallaschke. VI, 350 pages. 1985.

Vol. 256: Convexity and Duality in Optimization. Proceedings, 1984. Edited by J. Ponstein. V, 142 pages. 1985.

Vol. 257: Dynamics of Macrosystems. Proceedings, 1984. Edited by J.-P. Aubin, D. Saari and K. Sigmund. VI, 280 pages. 1985.

Vol. 258: H. Funke, Eine allgemeine Theorie der Polypol- und Oligopolpreisbildung. III, 237 pages. 1985.

Vol. 259: Infinite Programming. Proceedings, 1984. Edited by E. J. Anderson and A. B. Philpott. XIV, 244 pages. 1985.

Vol. 260: H.-J. Kruse, Degeneracy Graphs and the Neighbourhood Problem. VIII, 128 pages. 1986.

Vol. 261: Th. R. Gulledge, Jr., N. K. Womer, The Economics of Made-to-Order Production. VI, 134 pages. 1986.

Vol. 262: H. U. Buhl, A Neo-Classical Theory of Distribution and Wealth. V, 146 pages. 1986.

Vol. 263: M. Schäfer, Resource Extraction and Market Structure. XI, 154 pages. 1986.

Vol. 264: Models of Economic Dynamics. Proceedings, 1983. Edited by H.F. Sonnenschein. VII, 212 pages. 1986.

Vol. 265: Dynamic Games and Applications in Economics. Edited by T. Başar. IX, 288 pages. 1986.

Vol. 266: Multi-Stage Production Planning and Inventory Control. Edited by S. Axsäter, Ch. Schneeweiss and E. Silver. V, 264 pages. 1986.

Vol. 267: R. Bemelmans, The Capacity Aspect of Inventories. IX, 165 pages. 1986.

Vol. 268: V. Firchau, Information Evaluation in Capital Markets. VII, 103 pages. 1986.

Vol. 269: A. Borglin, H. Keiding, Optimality in Infinite Horizon Economies. VI, 180 pages. 1986.

Vol. 270: Technological Change, Employment and Spatial Dynamics. Proceedings 1985. Edited by P. Nijkamp. VII, 466 pages. 1986.

Vol. 271: C. Hildreth, The Cowles Commission in Chicago, 1939–1955. V, 176 pages. 1986.

Vol. 272: G. Clemenz, Credit Markets with Asymmetric Information. VIII, 212 pages. 1986.

Vol. 273: Large-Scale Modelling and Interactive Decision Analysis. Proceedings, 1985. Edited by G. Fandel, M. Grauer, A. Kurzhanski and A.P. Wierzbicki. VII, 363 pages. 1986.

Vol. 274: W.K. Klein Haneveld, Duality in Stochastic Linear and Dynamic Programming. VII, 295 pages. 1986.

Vol. 275: Competition, Instability, and Nonlinear Cycles. Proceedings, 1985. Edited by W. Semmler. XII, 340 pages. 1986.

Vol. 276: M.R. Baye, D.A. Black, Consumer Behavior, Cost of Living Measures, and the Income Tax. VII, 119 pages. 1986.

Vol. 277: Studies in Austrian Capital Theory, Investment and Time. Edited by M. Faber. VI, 317 pages. 1986.

Vol. 278: W.E. Diewert, The Measurement of the Economic Benefits of Infrastructure Services. V, 202 pages. 1986.

Vol. 279: H.-J. Büttler, G. Frei and B. Schips, Estimation of Disequilibrium Models. VI, 114 pages. 1986.

Vol. 280: H.T. Lau, Combinatorial Heuristic Algorithms with FORTRAN. VII, 126 pages. 1986.

Vol. 281: Ch.-L. Hwang, M.-J. Lin, Group Decision Making under Multiple Criteria. XI, 400 pages. 1987.

Vol. 282: K. Schittkowski, More Test Examples for Nonlinear Programming Codes. V, 261 pages. 1987.

Vol. 283: G. Gabisch, H.-W. Lorenz, Business Cycle Theory. VII, 229 pages. 1987.

Vol. 284: H. Lütkepohl, Forecasting Aggregated Vector ARMA Processes. X, 323 pages. 1987.

Vol. 285: Toward Interactive and Intelligent Decision Support Systems. Volume 1. Proceedings, 1986. Edited by Y. Sawaragi, K. Inoue and H. Nakayama. XII, 445 pages. 1987.

Vol. 286: Toward Interactive and Intelligent Decision Support Systems. Volume 2. Proceedings, 1986. Edited by Y. Sawaragi, K. Inoue and H. Nakayama. XII, 450 pages. 1987.

Vol. 287: Dynamical Systems. Proceedings, 1985. Edited by A.B. Kurzhanski and K. Sigmund. VI, 215 pages. 1987.

Vol. 288: G.D. Rudebusch, The Estimation of Macroeconomic Disequilibrium Models with Regime Classification Information. VII, 128 pages. 1987.

Vol. 289: B.R. Meijboom, Planning in Decentralized Firms. X, 168 pages. 1987.

Vol. 290: D.A. Carlson, A. Haurie, Infinite Horizon Optimal Control. XI, 254 pages. 1987.

Vol. 291: N. Takahashi, Design of Adaptive Organizations. VI, 140 pages. 1987.

Vol. 292: I. Tchijov, L. Tomaszewicz (Eds.), Input-Output Modeling. Proceedings, 1985. VI, 195 pages. 1987.

Vol. 293: D. Batten, J. Casti, B. Johansson (Eds.), Economic Evolution and Structural Adjustment. Proceedings, 1985. VI, 382 pages. 1987.

Vol. 294: J. Jahn, W. Krabs (Eds.), Recent Advances and Historical Development of Vector Optimization. VII, 405 pages. 1987.

Vol. 295: H. Meister, The Purification Problem for Constrained Games with Incomplete Information. X, 127 pages. 1987.

Vol. 296: A. Börsch-Supan, Econometric Analysis of Discrete Choice. VIII, 211 pages. 1987.

Vol. 297: V. Fedorov, H. Läuter (Eds.), Model-Oriented Data Analysis. Proceedings, 1987. VI, 239 pages. 1988.

Vol. 298: S.H. Chew, Q. Zheng, Integral Global Optimization. VII, 179 pages. 1988.

Vol. 299: K. Marti, Descent Directions and Efficient Solutions in Discretely Distributed Stochastic Programs. XIV, 178 pages. 1988.

Vol. 300: U. Derigs, Programming in Networks and Graphs. XI, 315 pages. 1988.

Vol. 301: J. Kacprzyk, M. Roubens (Eds.), Non-Conventional Preference Relations in Decision Making. VII, 155 pages. 1988.

Vol. 302: H.A. Eiselt, G. Pederzoli (Eds.), Advances in Optimization and Control. Proceedings, 1986. VIII, 372 pages. 1988.

Vol. 303: F.X. Diebold, Empirical Modeling of Exchange Rate Dynamics. VII, 143 pages. 1988.

Vol. 304: A. Kurzhanski, K. Neumann, D. Pallaschke (Eds.), Optimization, Parallel Processing and Applications. Proceedings, 1987. VI, 292 pages. 1988.

T. Vasko (Ed.)

The Long-Wave Debate

Selected papers from an IIASA (International Institute for Applied Systems Analysis) International Meeting on Long-Term Fluctuations in Economic Growth: Their Causes and Consequences, Held in Weimar, German Democratic Republic, June 10–14, 1985

1987. 128 figures. XVII, 431 pages. ISBN 3-540-18164-4

Contents: Concepts and Theories on the Interpretation of Long-Term Fluctuations in Economic Growth. - Technical Revolutions and Long Waves. - The Role of Financial and Monetary Variables in the Long-Wave Context. - Modeling the Long-Wave Context. - Modeling the Long-Wave Phenomenon. - List of Participants.

I. Boyd, J. M. Blatt

Investment Confidence and Business Cycles

1988. 160 pages. ISBN 3-540-18516-X

Contents: Introduction and brief summary. - A brief historical survey of the trade cycle. - Literature on confidence. - The dominant theories. - A first look at the new model. - Confidence. - Description of the model. - The longer run. - Some general remarks. - Appendices. - References. - Index.

M. Faber, H. Niemes, G. Stephan

Entropy, Environment and Resources

An Essay in Physico-Economics

With the cooperation of L. Freytag

Translated from the German by I. Pellengahr

1987. 33 figures. Approx. 210 pages. ISBN 3-540-18248-9

The special features of the book are that the authors utilize a natural scientific variable, entropy, to relate the economic system and the environment, that environmental protection and resource use are analyzed in combination, and that a replacement of techniques over time is analyzed. A novel aspect is that resource extraction is interpreted as a reversed diffusion process. Thus a relationship between entropy, energy and resource concentration is established.

E. van Damme

Stability and Perfection of Nash Equilibria

1987. 105 figures. Approx. 370 pages. ISBN 3-540-17101-0

Contents: Introduction. - Games in Normal Form. - Matrix and Bimatrix Games. - Control Costs. - Incomplete Information. - Extensive Form Games. - Bargaining and Fair Division. - Repeated Games. - Evolutionary Game Theory. - Strategic Stability and Applications. - References. - Survey Diagrams. - Index.

Springer-Verlag
Berlin Heidelberg NewYork
London Paris Tokyo